Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jure Malovrh

Prenos podatkov z ultrazvokom

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Matija Marolt

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu preučite področje prenosa podatkov preko ultrazvoka oz. človeku neslišnih frekvenc. Izdelajte sistem za tovrsten prenos, ki bo uporabljal različne načine kodiranja informacij in preizkusite njegovo hitrost in zanesljivost pri različnih pogojih, kot so: vrsta naprav, jakost šuma v okolici in oddaljenost med napravami.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jure Malovrh sem avtor diplomskega dela z naslovom:

Prenos podatkov z ultrazvokom

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Matije Marolta,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 3. september 2015 Podpis avtorja:

Zahvaljujem se mentorju, doc. dr. Matiji Maroltu, za vso potrpežljivost, pomoč, sestanke in nasvete pri izdelavi naloge. Zahvaljujem se tudi asistentu Matevžu Pesku, ki mi je bil vedno na voljo za pomoč in razlage. Posebna zahvala gre tudi moji družini za podporo pri študiju. Na koncu bi se rad zahvalil še vsem prijateljem in kolegom na fakulteti, ki so mi tekom študija pomagali in ga naredili zabavnejšega.

Družini

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
FT	Fourier transform	Fourierova transformacija
DFT	Discrete Fourier transform	diskretna Fourierova transformacija
FFT	Fast Fourier transform	hitra Fourierova transformacija
STMF	Single-tone multi-frequency signaling	monotonska mnogofrekvenčna signalizacija
H(7,4)	Hamming code (7,4)	Hammingov kod (7,4)
HP filter	high pass filter	visokofrekvenčni filter
P2P	peer to peer	omrežje vsak z vsakim
P2MP	point to multipoint	omrežje eden z mnogimi
NFC	near field communication	komunikacija kratkega dosega

Povzetek

V pričujočem delu se ukvarjamo s področjem prenosa podatkov preko neslišnih frekvenc. V ta namen smo razvili Android aplikacijo, ki je namenjena pošiljanju in prejemanju zvoka. Za namen testiranja prenosa podatkov smo razvili tri različne načine prenosa, ki smo jih med seboj natančno primerjali. Naš cilj je bil prenos 32 bitov v času od dveh do treh sekund. Vsako metodo smo predstavili z več podmetodami, ki so se med seboj razlikovale v hitrosti prenosa. Na ta način smo med njimi prikazali odvisnost med hitrostjo prenosa in natančnostjo prejemanja. Cilj smo uspešno dosegli z vsemi tremi metodami.

Ključne besede: prenos podatkov, ultrazvok, Android.

Abstract

In our work we are focusing on data transmission via inaudible frequencies. For this we have developed an Android application that is designed to send and receive sound. For the purpose of transmission tests we have developed three distinct methods of transmission, which have been compared to each other. In our tests we took into account several criteria, among which we wanted to find correlations. Our goal is the transfer of 32 bits in the shortest time possible. Each method was presented with several submethods, which differed in the speed of transmission. With submethods we show the correlation between transmission speed and precision of receiving. We were successful in achieving our goal with all of the methods.

Keywords: data transmission, ultrasound, Android.

Poglavje 1

Uvod

V današnjem času se število elektronskih naprav povečuje s skoraj eksponentno hitrostjo. Problem, ki nastane zaradi velikega števila naprav, je komunikacija med njimi. Težave s komunikacijo lahko delno rešujemo s pomočjo interneta, uporabo bluetooth-a, NFC-ja idr., vendar pa to v primeru preprostejših naprav ne pride v poštev. Ena izmed možnosti je prenos podatkov s pomočjo zvoka. Zvočnik je namreč precej manjši od drugih modulov in tudi lažje implementiran. S pomočjo zvoka bi tako lahko obšli omejitve, ki v današnjih časih nastajajo zaradi različnih standardov za povezovanje med napravami in različne strojne ali programske opreme (npr., ena mobilna naprava z NFC-jem in druga brez, različne verzije bluetooth-a, ni dostopa do internetne povezave, različni operacijski sistemi). Naprave, ki so narejene po meri človeka, imajo navadno svoj razpon določen med 20 Hz in 20 kHz. Slišimo torej zvok, ki zaniha med 20-krat do 20000-krat na sekundo, kolikor je v povprečju maksimalen slušni razpon mladega in zdravega človeka [1]. Vendar pa se z leti, glasnim okoljem in obrabo naš slušni razpon znižuje, zato velika večina ljudi dosega najvišjo slušno frekvenco med 16 in 17 kHz. Torej imamo na vsaki napravi, ki ima zvočnik, med tri in štiri kHz manevrskega prostora, kjer obstajajo frekvence, ki jih ljudje ne slišijo, naprave za ljudi pa jih lahko zaznajo. Tako lahko s pomočjo naprave, ki generira in pošilja zvok, ter naprave, ki zmore ta zvok prebrati, dobimo prenos podatkov,

ki je ljudem v večini primerov neslišen. Tak prenos podatkov ne omogoča tako velike hitrosti, kot jo omogočajo drugi mediji, zadostuje pa za prenos manjših, preprostejših podatkov.

Naslov naloge je sicer prenos podatkov z ultrazvokom, vendar pa je to le termin, ki opisuje našo nezmožnost slišanja frekvenc. Morda bi bil boljši izraz neslišne frekvence, saj jih ljudje v večini primerov ne slišimo, vendar pa te ne spadajo v območje ultrazvoka, ki se začne nad našo slišno frekvenco, t.j. 20kHz.

V nalogi se torej ukvarjamo s prenosom podatkov preko ultrazvoka. V ta namen smo razvili mobilno aplikacijo na sistemu Android, ki je namenjena prejemanju in pošiljanju zvoka. Cilj naše naloge je bil doseči hitrost prenosa podatkov okoli deset bitov na sekundo, natančneje, doseči smo želeli prenos 32 bitov v dveh do treh sekundah.
Pri tem smo preizkusili različne načine prenašanja podatkov. Preizkusili smo **binarno** prenašanje, kjer sta dovolj visoki frekvenci predstavljali našo ničlo in enico; **STMF** prenašanje, kjer smo abecedo razdelili na celoten spekter in jo nato pošiljali znak po znak ter **spektralno** pošiljanje, kjer smo naenkrat pošiljali spekter osmih znakov.
Za varnost podatkov je pri binarnem in spektralnem pošiljanju poskrbljeno s Hammingovim kodom (7, 4), ki našim štirim podatkovnim bitom doda tri varnostne bite. Dodatno smo mu dodali še pariteto nad celotnim, že zgeneriranim paketom, tako da lahko popravljamo enojno in detektiramo dvojno napako. Vse tri metode smo preizkusili v različnih okoliščinah, pri čemer smo ugotavljali, kolikšna je najvišja bitna hitrost, ki jo lahko dosežemo, ter kolikšen delež poslanih paketov je bil poslan pravilno.
Našo aplikacijo bi lahko uporabljali povsod, kjer je na voljo vir, ki lahko oddaja zvok. Na tak način bi lahko npr. trgovci bolj natančno opredelili našo lokacijo in nam glede na njo pošiljali ponudbe.

V naslednjem poglavju bomo pregledali področje, ki je osnova pričujočega diplomskega dela. Funkcije in metode, ki smo jih pri implementaciji uporabili, bomo natančneje pregledali v tretjem poglavju. V četrtem poglavju

bomo pregledali metode, ki smo jih implemetirali sami. Rezultate in ovrednotenje naših testov bomo prikazali v predzadnjem poglavju, kjer bomo natančneje pregledali, kakšne rezultate smo dosegli in podali primere, kako bi jih lahko še izboljšali. V zaključku pa bomo naše delo povzeli in preverili, ali so bili cilji doseženi.

Poglavje 2

Pregled področja

Kaj pravzaprav prenos podatkov sploh pomeni? Prenos podatkov se nanaša na fizični prenos podatkov skozi komunikacijski kanal. Komunikacijske kanale delimo na P2P ali P2MP, podatke pa se lahko pošilja sinhrono ali asinhrono.[2] Primeri komunikacijskih kanalov so bakrene žice, optična vlakna, brezžična komunikacija, bluetooth, NFC, različni pomnilni mediji in računalniške povezave. Poznamo torej precej načinov prenosa podatkov, vendar pa se v poplavi naprav, ki zbirajo ali na kakršen koli drug način pridobivajo podatke, pojavi problem, kako vse te naprave povezati med seboj, po kakšnem komunikacijskem kanalu naj torej podatke pošiljamo. In prav tu lahko svojo vlogo odigra zvok.

Zvok je valovanje v zraku, podobno valovanju, ki nastane, ko v vodo vržemo kamen. Nastane z vibracijo nekega predmeta. Hitrost vibracije predmeta bo določala frekvenco zvoka. Predmet, ki bo vibriral z nizko frekvenco, bo proizvedel nizek ton, predmet z vibracijo visoke frekvence pa bo proizvedel visok ton. Kot smo že omenili, se slušne frekvence človeka gibljejo med 20 Hz in 20 kHz, torej lahko slišimo predmete, ki vibrirajo od 20-krat do 20000-krat na sekundo (v teoriji; v praksi je ta številka okoli 16000).[3]

2.1 Prenos podatkov z zvokom

Prenos podatkov v slišni frekvenci se uporablja redko, saj je moteč za okolico in človeka. Zaradi posebnih lastnosti, pa se lahko uporablja v okolju, kjer njegova prisotnost ni tako moteča. Največkrat se tako uporablja v vodnem okolju, saj se zvočni valovi odlično širijo pod vodo in so zmožni prepotovati daljše razdalje v primerjavi z radio valovi, ki jih slana voda oslabi, ter optičnimi valovi, ki se v vodi razpršijo [4].

Na podlagi podvodnih avdio prenosov so se odločili projekt preizkusiti tudi pri podjetju Google, kjer so ustvarili komunikacijo med napravami z zvokom [5]. Naprave se med seboj povežejo z začetnim tonom, sledi kalibracija, ki poskrbi, da imata napravi enake nastavitve prejemanja, čemur sledi sinhrono prejemanje podatkov.

Za enostavnejši prenos podatkov preko mobilnega telefona so preizkusili prenos preko telefonske linije, ki deluje tako, da na eni strani telefon podatke pošilja, na drugi strani pa jih telefon sprejme in obdela [6]. Na tak način lahko enostavne podatke med pogovorom našemu sogovorniku pošljemo. Namesto, da bi se zapletali z zapletenimi navodili, kako priti do neke geografske točke, bi naslov enostavno zgenerirali kot serijo zvokov, ga poslali napravi na drugi strani, ki bi dekodirala in s pomočjo aplikacije poiskala navigacijo. To je le ena izmed možnosti.

Poznamo tudi podjetji, ki sta se s prenosom podatkov preko zvoka začeli ukvarjati komercialno. Podjetji Chirp [7] in Nearbytes [8] sta se problema lotili na različne načine, obema pa je skupno to, da za prenos podatkov uporabljata slišne frekvence zvoka.

Podjetje Chirp je prenos preko zvoka izpeljalo tako, da je za zvočni spekter vzelo vrednosti med 1760 Hz in 10.5 kHz. Posamezna črka je torej pomenila neko frekvenco na tem območju.

Podjetje Nearbytes je za svoj zvočni spekter uporabilo visoke frekvence, ki so še vedno v slišnem območju človeka. Ker so se osredotočali predvsem na varnost, so uporabljali višje frekvence, saj je razdalja, po kateri se visoka frekvenca porazgubi, precej nižja od razdalje nizke frekvence.

2.2 Uporaba ultrazvoka

O ultrazvoku govorimo, kadar presežemo mejo teoretično slišnega zvoka 20 kHz. Z ultrazvokom se podatkov običajno ne prenaša, saj se prehitro porazgubijo; delno se uporablja le v podmornicah, kjer poleg slišnih frekvenc občasno uporabljajo tudi ultrazvočne.

Dodatno bi lahko z ultrazvokom določali položaj v trgovinah ali ostalih zaprtih prostorih, kjer je uporaba GPS-a omejena. To metodo so poskušali razviti v Dublinu, kjer so uporabljali ultrazvočne frekvence med 20 in 22 kHz, kar je meja zmogljivosti na večini mikrofonov, ki so v pametnih telefonih [9]. S pomočjo ultrazvoka jim je uspelo pridobivati lokacijo na manj kot meter natančno, kar je v zaprtem prostoru zelo dober uspeh.

Drugi primeri uporabe ultrazvoka so bolj kot prenašanju podatkov namenjeni njihovemu zbiranju:

- **sonar**: pošiljanje signala v vse smeri, spremljanje odboja na poti, če se je odboj zgodil, smo zadeli oviro;
- **merjenje globine;**
- **medicina**: slikanje notranjih organov, zarodkov pri nosečnicah idr. in
- **različne terapije v športu in medicini.**

2.3 Prenos podatkov z neslišno frekvenco

Poskusi prenosa podatkov z neslišno frekvenco so v zadnjem času v velikem porastu. To lahko pripišemo dejstvu, da so pametni telefoni že tako vseprisotni, da si ljudje in podjetja želijo izkoristiti njihove prednosti. Prejemanje neslišne frekvence nas ne stane nič, saj ne potrebujemo nobenih posebnih dodatkov v telefonu, ker sta mikrofon in zvočnik že vgrajena. [10, 11, 12, 13, 14]

Iz vseh poskusov prenosa podatkov smo poskusili izluščiti čim več, kar bi nam omogočilo lažje in bolj učinkovite metode za reševanje naše težave. Naše raziskovanje smo začeli z aplikacijo Notifly [14]. Aplikacija deluje po

principu spektra in ker smo pri implementaciji podobno metodo uporabili tudi mi, je spekter podrobneje predstavljen v naslednjem poglavju.

Boris Smus [11] je celotno aplikacijo in njeno delovanje prestavil v splet, pri čemer se je opiral predvsem na delovanje preko programskega jezika JavaScript. Metoda, ki jo je uporabljal, je STMF, ki je natančneje predstavljena v naslednjem poglavju.

Rusko podjetje Azoft [12] je predstavilo svojo metodo pošiljanja podatkov kot alternativo NFC-ju. Pri pošiljanju podatkov so se zanašali le na ničlo in enico, saj so pošiljali binarne podatke. Podobno metodo smo uporabili tudi mi, zato več o njej sledi v naslednjem poglavju. Pri prebiranju podatkov so namesto FFT-ja, ki smo ga uporabili mi, uporabili FIR filter.

Podjetje Signal360 [13] je edino izmed podjetij, ki jim pošiljanje podatkov predstavlja poslovno dejavnost. Glede njihovih metod na spletu ni veliko znanega, za svoje delovanje pa uporabljajo t.i *beacone* oz. po slovensko žolne, ki oddajajo zvok in tako posredujejo podatke na aplikacijo.

Poglavje 3

Metode in orodja

V tem poglavju sledi predstavitev pomembnejših metod in orodij, ki smo jih uporabljali pri naši implementaciji.

Pri vsaki implementaciji, ki ima kolikor toliko opraviti z zvokom in njegovimi frekvencami, bomo srečali Fourierovo transformacijo, kjer bomo predstavili osnovno DFT in njeno naprednejšo različico FFT. Ker se v primeru, da ne zajamemo celotne periode signala vanj vnaša šum, bomo spoznali okenske funkcije, ki poskušajo napake odstraniti. Predstavljen je tudi varnostni vidik našega pošiljanja, zagotovljen s Hammingovim kodom (7,4), ki smo ga dodatno zavarovali za zaznavanje dvojnih napak.

3.1 Fourierova transformacija

Fourierova transformacija [15] je transformacija, ki funkcijo, odvisno od časa, preslika v frekvence, ki jo sestavljajo. Gre torej za preslikavo iz časovnega v frekvenčni prostor. Fourierova transformacija deluje na obeh vrstah signalov, na zveznih (časovna komponenta ni določena, funkcija je neštevna), kjer imamo težave s predstavitvijo popolnoma kvadratne funkcije, ter diskretnih (določena časovna komponenta, funkcija je števna). Ker se v našem primeru ukvarjamo z zvokom, ki ga vzorčimo z določeno frekvenco (kar pomeni da imamo števno število vzorcev), je naša Fourierova transformacija popolnoma

določena, zato uporabljamo posebno podvrsto transformacije, imenovano diskretna Fourierova transformacija.

3.1.1 Diskretna Fourierova transformacija

Kot že omejeno, diskretna Fourierova transformacija temelji na sekvenci števil (običajno pridobljenih z merjenjem nečesa), ki jim želimo izmeriti frekvenco. Splošna enačba za transformacijo DTF (3.1) nam iz vsakega števila, ki ga podamo v sekvenci dolžine N, izračuna diskretne koeficiente. To da v enačbi morda ne vidimo sinusoide, je posledica tega, da je sinusoida zapisana v eksponentni obliki (3.2). Če funkcijo dodatno transformiramo v preglednejšo obliko, dobimo končno in precej lepšo obliko DFT, ki izgleda precej enostavno (3.3). Končna oblika DFT-ja nam pove korelacijo med kosinusom in sinusom.

Izrek 3.1 *Formula za transformacijo*

$$X_k = \sum_{n=0}^{N-1} x_n * e^{-2\pi ikn/N}, k \in \mathbb{Z} \tag{3.1}$$

Izrek 3.2 *Eksponentna oblika sinusoide*

$$e^{-i\theta} = \cos\theta - i\sin\theta \rightarrow \theta = 2\pi kn/N \rightarrow X_k = \sum_{n=0}^{N-1} x_n * (\cos(\frac{2\pi kn}{N}) - i\sin(\frac{2\pi kn}{N})) \tag{3.2}$$

Izrek 3.3 *Končna oblika DFT transformacije*

$$X_k = \sum_{n=0}^{N-1} x_n \cos(\frac{2\pi kn}{N}) - i\sum_{n=0}^{N-1} x_n \sin(\frac{2\pi kn}{N}) \tag{3.3}$$

Delovanje DFT algoritma je najlažje prikazati na enostavnem primeru (3.1), v katerem je $N = 4$ in za katerega bomo uporabili poenostavljeno enačbo (3.3). Končni rezultat, ki smo ga dobili, je $[60 - 0i, 12 + 8i, 12 - 0i, 12 - 8i]$.

Primer 3.1 *Primer delovanja DFT na enostavnem primeru* $x = [24, 8, 12, 16]$

$$\begin{aligned} k &= 0 \\ 24cos(0) + 8cos(0) + 12cos(0) + 16cos(0) - \\ i(24sin(0) + 8sin(0) + 12sin(0) + 16sin(0)) &= \\ (24 + 8 + 12 + 16) - 0i &= \quad 60 - 0i \end{aligned} \tag{3.4}$$

$$\begin{aligned} k &= 1 \\ 24cos(0) + 8cos(\tfrac{\pi}{2}) + 12cos(\pi) + 16cos(\tfrac{3\pi}{2}) - \\ i(24sin(0) + 8sin(\tfrac{\pi}{2}) + 12sin(\pi) + 16sin(\tfrac{3\pi}{2})) &= \\ (24 - 12) - (8 - 16)i &= \quad 12 + 8i \end{aligned} \tag{3.5}$$

$$\begin{aligned} k &= 2 \\ 24cos(0) + 8cos(\pi) + 12cos(2\pi) + 16cos(3\pi) - \\ i(24sin(0) + 8sin(\pi) + 12sin(2\pi) + 16sin(3\pi)) &= \\ (24 - 8 + 12 - 16) - (0)i &= \quad 12 - 0i \end{aligned} \tag{3.6}$$

$$\begin{aligned} k &= 3 \\ 24cos(0) + 8cos(\tfrac{3\pi}{2}) + 12cos(3\pi) + 16cos(\tfrac{9\pi}{2}) - \\ i(24sin(0) + 8sin(\tfrac{3\pi}{2}) + 12sin(3\pi) + 16sin(\tfrac{9\pi}{2})) &= \\ (24 - 12) - (-8 + 16)i &= \quad 12 - 8i \end{aligned} \tag{3.7}$$

$$X = [60 - 0i, 12 + 8i, 12 - 0i, 12 - 8i] \tag{3.8}$$

Ker DFT deluje v obe smeri, lahko naše podatke preslikamo iz frekvenčne domene v časovno s pomočjo enačbe (3.9). V naši aplikaciji tega ne potrebujemo. Je pa inverznost enačbe pomemben del veliko aplikacij, ki stvari enostavneje izračunavajo v frekvenčni domeni, nato pa rezultate zgolj preslikajo nazaj v časovno domeno.

Edina slaba lastnost DFT je njena hitrost, saj vidimo, da se moramo za vsak k, ki je dolžine n, sprehoditi čez celoten seznam, ki je prav tako dolžine n. Časovna zahtevnost takega programa je torej $O(n^2)$, kar pomeni, da časovna zahtevnost raste eksponentno z dolžino seznama. To je bil tudi razlog za poskus pohitritve metode, kar je uspelo z metodo FFT, ki je natančneje predstavljena v naslednjem podpoglavju.

$$x_n = \frac{1}{n} \sum_{k=0}^{N-1} X_k e^{i2\pi kn/N} \tag{3.9}$$

3.1.2 Hitra Fourierova transformacija

Hitra Fourierova transformacija je bila razvita z namenom hitrejšega delovanja DFT, ki je bila za delovanje v interaktivnih aplikacijah prepočasna. V ta namen so se raziskovalci dolga leta ukvarjali z vprašanjem, kako pohitriti DFT. Avtorja, ki se smatrata kot avtorja generične verzije FFT-ja [16], sta v letu 1965 izdala članek, ki je opisoval generično metodo FFT, ki pri svojem delovanju ni bila omejena z dolžino sekvence. Od takrat naprej je FFT nepogrešljiv v velikem številu aplikacij, saj je omogočil hitro izračunavanje sekvenc pri transformaciji iz časovnega ali prostorskega spektra v frekvenčni spekter.

Kako FFT pravzaprav deluje? [17, 15] V našem primeru smo predpostavljali, da je prejeti signal dolžine potence števila dve. FFT deluje, tudi če to ni tako, in sicer tako da doda ničle do dolžine najbližje potence števila dve. Signal, ki ga prejmemo, bomo delili na več delov, po metodi deli in vladaj. To je rekurzivna metoda. Celoten FFT izvedemo v treh korakih:

1. V prvem koraku naše podatke rekurzivno delimo na polovico. To delamo, dokler ni dolžina posameznega seznama enaka ena. Deljenje na polovice prikazuje slika 3.1.

2. Po deljenju moramo izračunati frekvenčni spekter vseh točk. To je v primeru ene točke trivialno, zato se enostavno prestavimo iz časovnega v frekvenčni prostor.

3. Zadnji korak je združevanje posameznih točk nazaj v seznam točk. Primer združevanja je prikazan na sliki 3.2, kjer na levi strani, pri področju Time domain, opazujemo, kako se združuje nazaj v skupen seznam. Vidimo lahko, da se posamezne prostore zapolni z izračunano

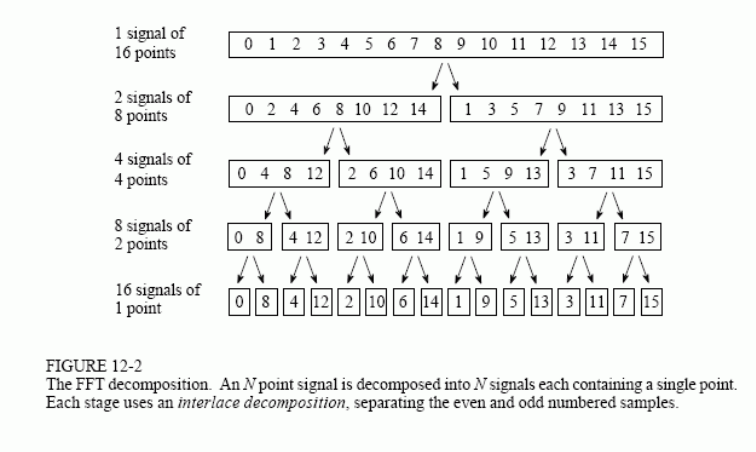


FIGURE 12-2
The FFT decomposition. An N point signal is decomposed into N signals each containing a single point. Each stage uses an *interlace decomposition*, separating the even and odd numbered samples.

Slika 3.1: Deljenje prejetega signala, vir slike: http://www.dspguide.com/ch12/2.htm

vrednostjo ali pa z 0 v primeru, da smo spekter izračunali na drugi polovici. Na primeru 3.2 to pomeni, da se $a0b0c0d0$ in $0e0f0g0h$ združita skupaj in tvorita sestavljen izračun $aebfcgdh$. Tako nadaljujemo, dokler ne pridemo nazaj do začetnega stanja, kjer imamo sedaj v seznamu zapisan frekvenčni spekter.

Zakaj je torej FFT tako pomemben? Če pogledamo število potrebnih deljenj, vidimo, da potrebujemo $log_2 N$ ponovitev. Seštevanj, ki jih moramo izvesti v vsaki ponovitvi, je N ($N/2$ za vsako stran); celoten FFT se torej izvede v $O(Nlog_2 N)$ [18]. Na prvi pogled ta pospešitev ne izgleda nič posebnega, vendar že preprost izračun pri manjšem, 1024 zapisov velikem seznamu pokaže: $1024^2 = 1048576$ operacij proti $1024log_2 1024 = 10240$. Torej je FFT $1048576/10240 = 102.40$-krat hitrejši. Ta razlika pa se z večanjem N le še povečuje.

3.2 Okenske funkcije

Pri zajemanju signala naša FT pričakuje, da se bo v vzorcu nahajala celotna perioda zajetega signala. Vendar pa je v realnosti to redko tako, zato se nam

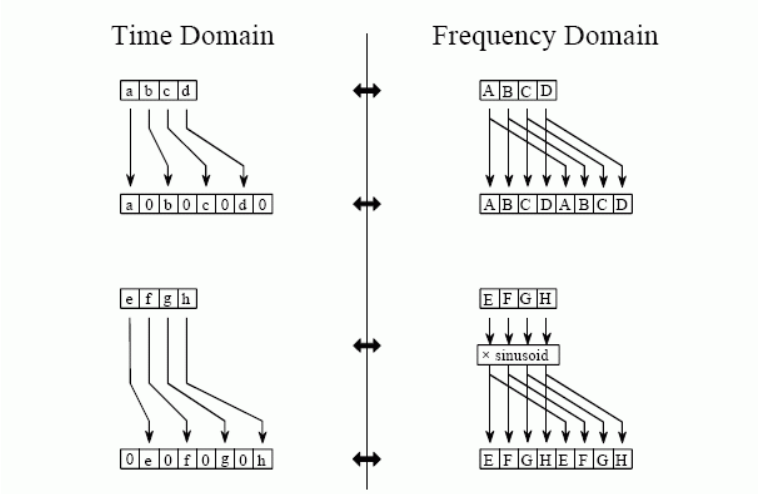


FIGURE 12-4
The FFT synthesis. When a time domain signal is diluted with zeros, the frequency domain is duplicated. If the time domain signal is also shifted by one sample during the dilution, the spectrum will additionally be multiplied by a sinusoid.

Slika 3.2: Združevanje izračunanega frekvenčnega prostora, vir slike: http://www.dspguide.com/ch12/2.htm

v vzorcu pojavi šum; frekvence, ki v signalu niso prisotne. Okenska funkcija nam pomaga odstraniti napake (spektralno uhajanje), vendar so zaradi tega vrhovi FT v signalu manj vidni, zato je pomembno, da so signali dovolj močni, da jih bomo kljub utišanju še vedno zaznali.

V našem delu smo preizkusili tri različne okenske funkcije, vsaka bo natančneje predstavljena in definirana v manjšem podpoglavju.

3.2.1 Generalizirano Hammingovo okno

Generalizirano Hammingovo okno ima formulo oblike (3.10), kjer je N dolžina seznama našega signala. Glede na vrednosti α in β ločimo dve različni podvrsti: Hannovo okno, ki ga velikokrat imenujejo tudi Hannigovo okno, kjer sta $\alpha == \beta$, torej imata oba isto vrednost 0.5, in Hammingovo okno, ki je optimizirano, da minimizira šum, ki je naši najvišji frekvenci najbližje. α in β imata tu vrednosti $\alpha = 0.54, \beta = 0.46$.

$$w(n) = \alpha - \beta cos(\frac{2\pi n}{N-1}) \tag{3.10}$$

3.2.2 Blackman–Nuttallovo in Blackman–Harrisovo okno

Okni sta si po sestavi precej podobni, saj uporabljata isto matematično formulo (3.11). Njuna razlika je le v vrednosti koeficientov a_0, a_1, a_2, a_3. Blackman–Nuttallovo uporablja vrednosti koeficientov $a_0 = 0.3635819, a_1 = 0.4891775, a_2 = 0.1365995$ in $a_3 = 0.0106411$. Blackman–Harrisovo okno nam šum minimizira bolje kot Blackman–Nuttallovo, z le manjšimi spremembami v vrednosti koeficientov $a_0 = 0.35875, a_1 = 0.48829, a_2 = 0.14128$ in $a_3 = 0.01168$. Natančnejšo tabelo z različnimi možnostmi vrednosti keoficientov lahko najdete v knjigi [19] na strani 151.

$$w(n) = a_0 + a_1 cos(\frac{2\pi n}{N}) + a_2 cos(\frac{4\pi n}{N}) + a_3 cos(\frac{6\pi n}{N}) \tag{3.11}$$

3.3 Hammingov kod

Pri pošiljanju podatkov se lahko zgodi, da pride do napake. Če bi podatke pošiljali brez vsake varnosti, sprejemnik na drugi strani ne bi vedel, ali je pri pošiljanju prišlo do napake ali ne. Napake so lahko dveh vrst. Prva vrsta je samostojna napaka, to je napaka, ko se v poslanem sporočilu spremeni ali izgubi le en samostojen bit. Pri drugi vrsti napake se zgodi cela serija napak, napake "izbruhnejo"(angl. *burst error*). Te napake so veliko težje obvladljive; težko je namreč opaziti, da je prišlo do napake, saj je bilo narobe poslanih veliko podatkov. V pričujočem delu smo se omejili na samostojne napake, torej na napake, kjer je bil spremenjen en bit.

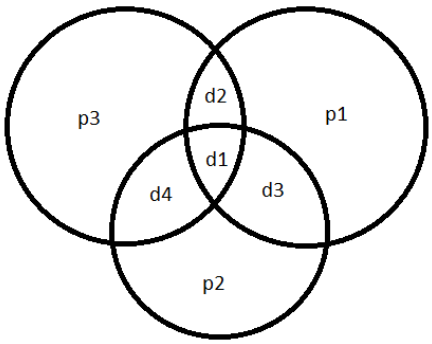
V ta namen smo uporabljali Hammingov kod(7,4) [20], ki štirim podatkovnim bitom doda tri varovalne, paritetne bite. Poleg treh dodatnih bitov lahko dodamo še en varovalni bit, ki je namenjen preverjanju paritete celotnega poslanega podatka. S pomočjo te paritete lahko zaznavamo, ali se je pri prenosu zgodila dvojna napaka in je v tem primeru na popravljamo. Zakaj napake ne popravljamo, je razloženo v nadaljevanju.

Gradnjo Hamminga je najenostavneje pokazati preko primera. Pri izgradnji našega Hammingovega koda smo uporabljali malce modificirano različico, ki vse paritetne bite doda na konec. To smo storili zaradi lažje izgradnje in dekodiranja prejete sekvence. Pri prikazu, katera pariteta kaže na določene podatke, si bomo pomagali s sliko 3.3, ki prikazuje Vennov diagram H(7,4). Primer, kako izgraditi osnoven H(7,4) (3.2), bo pokazal, kako lahko se tak kod zgradi.

Primer 3.2 *Primer izgradnje koda H(7,4).*

Želimo zgraditi H(7,4) iz podatkov 0110. Kot lahko vidimo na sliki 3.3, so paritete (biti označeni s črko p) odvisne od podatkov, ki jih želimo poslati. Pariteta doda toliko enic, da je njihovo skupno število sodo. Tako pariteto se običajno izračuna s pomočjo ekskluzivnega ali oz. XOR.

Na diagramu torej lahko opazimo, da pariteto $p1$ *dobimo iz podatkov* $d1$,$d2$ *in* $d3$*, pariteto* $p2$ *iz podatkov* $d1$*,* $d3$ *in* $d4$ *in pariteto* $p3$ *iz podatkov* $d1$*,* $d2$



Slika 3.3: Vennov diagram Hammingovega koda (7,4)

in $d4$.

$$p1 = d1 \oplus d2 \oplus d3 = 0 \oplus 1 \oplus 1 = 0 \tag{3.12}$$

$$p2 = d1 \oplus d3 \oplus d4 = 0 \oplus 1 \oplus 0 = 1 \tag{3.13}$$

$$p3 = d1 \oplus d2 \oplus d4 = 0 \oplus 1 \oplus 0 = 1 \tag{3.14}$$

Paket, ki ga pošljemo, je torej 0110011.

Podatek, ki ga pošljemo, je torej varnejši, saj bomo v primeru napake vedeli, da se je zgodila. Pri izračunavanju napake najprej izračunamo sindrom. Sindrom je izračunana vrednost naših poslanih podatkov in paritet. Logično je, da mora biti sindrom 0, če nismo prejeli nobene napake. Vendar pa v primeru, da smo napako prejeli, uporabimo matriko, ki nam pove, na katerem položaju je bila napaka. To matriko običajno imenujemo matrika za

preverjanje paritet P. V našem primeru izgleda tako:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Naš sindrom torej predstavlja indeks, kjer se je napaka zgodila. Če je sindrom enak 111, potem vemo, da se je napaka zgodila na prvem mestu (3.3).

Primer 3.3 *Prejeli smo podatke b = 1110011, napako vidimo na prvem mestu. Izračunali bomo vse tri sindrome in poskusili odpraviti napako.*

$$s1 = b1 \oplus b2 \oplus b3 \oplus b5 = 1 \oplus 1 \oplus 1 \oplus 0 = 1 \tag{3.15}$$

$$s2 = b1 \oplus b3 \oplus b4 \oplus b6 = 1 \oplus 1 \oplus 0 \oplus 1 = 1 \tag{3.16}$$

$$s3 = b1 \oplus b2 \oplus b4 \oplus b7 = 1 \oplus 1 \oplus 0 \oplus 1 = 1 \tag{3.17}$$

Sindrom, ki ga prejmemo, je torej 111; v matriki P pogledamo, na katerem mestu se ta sindrom nahaja in spremenimo bit na tistem mestu. Popravljen paket, ki smo ga popravili na prvem mestu, je 0110011, zapisano brez varnostih bitov smo prejeli 0110.

Zakaj pa nam problem predstavlja dvojna napaka? Ker H(7,4) ne ve, kdaj se je zgodila enojna in kdaj dvojna napake, bo v primeru dvojne napake poskusil popraviti bit na napačnem mestu (3.4).

Primer 3.4 *Prejeli smo podatke b = 1110111, napako vidimo na prvem in petem mestu. Izračunali bomo vse tri sindrome in poskusili odpraviti napako.*

$$s1 = b1 \oplus b2 \oplus b3 \oplus b5 = 1 \oplus 1 \oplus 1 \oplus 1 = 0 \tag{3.18}$$

$$s2 = b1 \oplus b3 \oplus b4 \oplus b6 = 1 \oplus 1 \oplus 0 \oplus 1 = 1 \tag{3.19}$$

$$s3 = b1 \oplus b2 \oplus b4 \oplus b7 = 1 \oplus 1 \oplus 0 \oplus 1 = 1 \tag{3.20}$$

Sindrom, ki ga prejmemo je torej 011, v matriki P se ta sindrom nahaja na četrtem mestu. H(7,4) poskuša popraviti kod in nam vrne 1111111, kar je

sicer pravilen H(7,4) kod, vendar pa so podatki napačni in namesto dvojne imamo sedaj trojno napako.

Zaradi primerov dvojne napake dodamo poleg običajnih treh varnostnih bitov še enega, ki izračuna pariteto nad celotnim H(7,4). Nato pri prejemu paketa najprej preverimo, če je glavna pariteta napačna. Če je glavna pariteta napačna, vemo, da se je zgodila enojna napaka, zato izračunamo sindrome in popravimo napako. Če pa kaže, da je glavna pariteta pravilna, sindrom pa kaže napako, potem se nam je zgodila dvojna napaka, ki je ne popravljamo, saj bi iz nje naredili trojno.

Vidimo torej, na kakšne stvari je potrebno paziti pri prejemanju podatkov. Varnost naših podatkov pa nam je hitrost pošiljanja podatkov tudi prepolovila, saj sedaj za osem prejetih podatkov kar štiri odstranimo z namenom varnosti. Tako imamo preveč poslanih kar 50 % podatkov, vendar pa imamo na ta račun večjo varnost in možnost popravljanja napak.

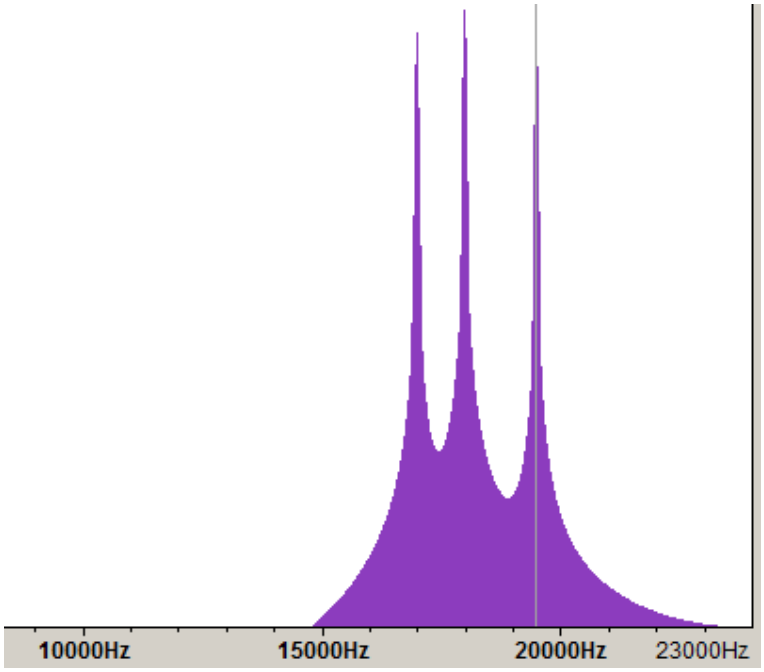
Poglavje 4

Implementacija

Za potrebe našega dela smo sprogramirali Android aplikacijo, ki je bila namenjena sprejemanju zvokov in njihovemu dekodiranju, kot tudi generiranju zvoka. Pri sprejemanju zvoka smo implementirali tri metode: binarno, STMF in spektralno, ki bodo natančneje predstavljene v nadaljevanju. Pri dekodiranju zvoka delujejo vse metode po podobnem principu. Zvok zaznamo in s pomočjo okenskih funkcij odstranimo napake (spektralno uhajanje). Signal nato obdelamo s FFT in glede na različne algoritme dekodiramo prejet zvok. Pri generiranju zvok generiramo na podlagi dolžine znaka, ki ga uporabljamo, in metode. Pri posameznem znaku smo dodali tudi postopno zviševanje glasnosti, kar nam pomaga, da je naš zvok bolj neslišen.

4.1 Binarno prejemanje podatkov

Prva metoda, ki smo jo implementirali, je binarno pošiljanje podatkov. Kako binarno pošiljanje podatkov deluje? V vseh primerih je začetni zvok enak 19.5 kHz. Ko naša naprava zazna začetni zvok, začne prejemati podatke. Med vsakim poslanim znakom sledi pavza. Na ta način programu povemo, da smo zaključili s pošiljanjem enega znaka in da bo na vrsti nov znak. Pavza je bila dodana zaradi možnosti pošiljanja istega znaka. Podatke naprava prejema, dokler ponovno ne zazna začetnega zvoka. S takim načinom asin-



Slika 4.1: Frekvenčni spekter binarnega pošiljanja podatkov

hronosti smo se izognili potrebam po začetni sinhronizaciji, saj smo vedeli, da bodo podatki prihajali, dokler ne bomo ponovno slišali našega končnega zvoka.

Za označitev ničle in enice smo izbrali frekvenci 17 in 18 kHz. Ti dve frekvenci sta bili izbrani, ker smo si želeli zagotoviti, da bi naša frekvenca ostala neslišna. Frekvenčni spekter je prikazan na sliki 4.1. Na sliki je vidno, da ima naš zvok najvišjo vrednost pri vseh treh prej omenjenih frekvencah: na 17 kHz, kjer zvok predstavlja enico, na 18 kHz, kjer zvok predstavlja ničlo ter na 19.5 kHz, kjer se nahajata začetni in končni zvok.

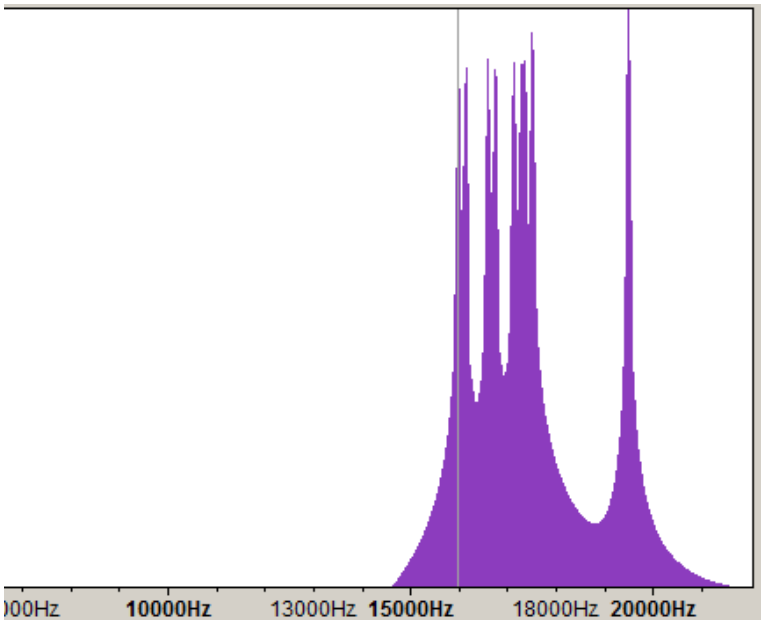
4.2 STMF prejemanje podatkov

Druga implementirana metoda je metoda STMF. Ime spominja na metodo DTMF, vendar pa ji ni podobna, saj namesto dveh frekvenc v našem primeru vzamemo le eno. Pri naši metodi vzamemo le eno frekvenco, ki nam

predstavlja našo črko, in jo pošljemo. Pri pošiljanju abecedo, ki jo lahko določimo sami, razdelimo na spekter, ki smo ga v tem primeru vzeli med 16 in 18 kHz. V tem primeru smo se spustili do meje naše slišne frekvence. To smo storili z namenom, da lahko pridobimo čim večji spekter za posamezno črko. V primeru, ko uporabljamo angleško abecedo, imamo skupno 26 črk in na posamezno črko pride približno 80 Hz, kar predstavlja manevrski prostor za eno črko. Omejili smo se na 18 kHz, saj se v primeru višanja te frekvence pogosto zgodi, da naš program namesto črke prebere zaključni znak in s tem prekine prejemanje podatkov, čeprav to še ni končano. Zgornjo mejo start/stop zvoka smo določili tako, da je še najbolj slišna napravi, saj se pri tako visokih frekvencah pogosto zgodi, da se na poti izničijo in naprava v tem primeru ne ve, da se je prenos že končal. Tako kot pri binarni metodi je med vsako posamezno črko pavza. Ta pavza morda izgleda nepotrebno, vendar imamo v angleščini precej primerov besed, ki vsebujejo dve skupni črki, zato je bilo pavzo potrebno dodati.
Frekvenčna slika testnega besedila, ki vsebuje črke K,P,C,U,I,U,A,R,S, je vidna na sliki 4.2. Na sliki morda izgleda, da so frekvence črk, ki nastopajo bolj skupaj, to so rs, us, pr, ac, idr., preveč skupaj, da bi jih lahko zaznali kot spremembo. Vendar je potrebno opozoriti, da pošiljamo posamezno frekvenco naenkrat, kar pomeni, da bi morala biti naša črka načeloma edina na tej frekvenci (v teoriji; v praksi lahko velikokrat iz okolja prejmemo šum). Problem nastane, če svojo abecedo povečamo na 40 ali 50 znakov, saj s tem povzročimo, da se več znakov preslika v eno frekvenco, kar povzroča kaos.

4.3 Spektralno prejemanje podatkov

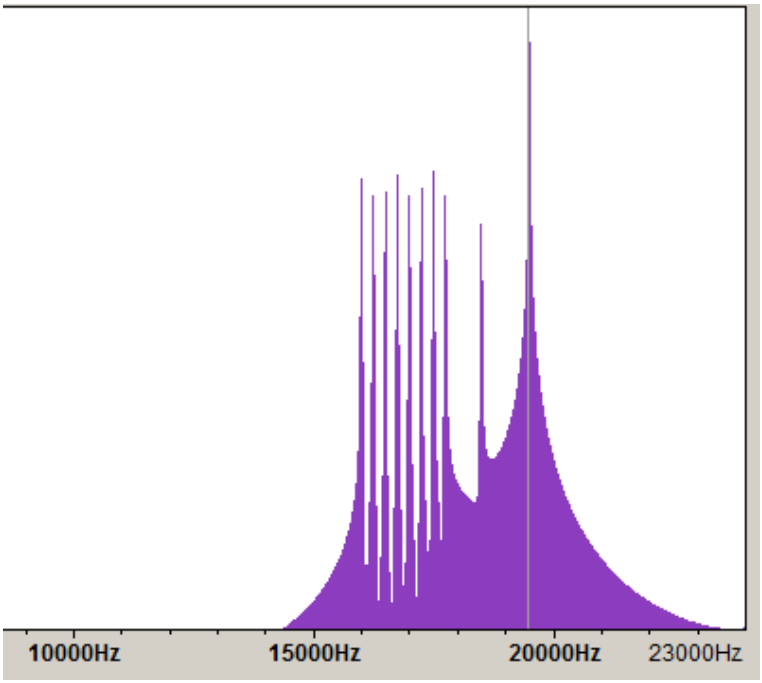
Pri obeh prejšnjih metodah smo videli, da je potrebno med posamezne znake dodajati pavzo, da prejemanje podatkov deluje kot mora. Toda pavza nam jemlje dragocen čas, ki bi ga lahko uporabili za prenos podatkov. Zato smo si zamislili idejo o spektralnem pošiljanju zvoka. V primeru spektralnega pošiljanja zvoka naenkrat pošljemo osem frekvenc, kjer prisotnost fre-



Slika 4.2: Frekvenčni spekter STMF pošiljanja podatkov

kvence označuje enico, odsotnost le-te pa ničlo. S tem smo se naenkrat res pošlje osem znakov, vendar pa to še vedno ne pomeni odsotnosti pavz med pošiljanjem. Zato smo poleg frekvenčnega spektra osmih znakov dodali še eno frekvenco, ki nam označuje uro. Ko se frekvenca ure obrne, vemo, da začnemo prejemati nov spekter znakov. Tako nam je uspelo iz našega pošiljanja odstraniti pavzo in s tem podvojiti hitrost pošiljanja. Ta podvojena hitrost pošiljanja pa je bila na koncu vseeno zmanjšana za polovico, saj smo štiri bite v podatkih uporabili za varnost.

Pri spektralnem pošiljanju (slika 4.3) imamo podatke razdeljene med 16 in 18 kHz, tako da vsaka izmed frekvenc zaseda 250 Hz. Ura se nahaja na frekvenci 18.5 kHz, start in stop zvok pa ostajata na 19.5 kHz. Težava, ki nastane pri spektralnem pošiljanju, je ta, da nam toliko frekvenc skupaj tvori nadležen ton, ki ga lahko slišimo. To je problem vzbujanja nižjih harmonikov. Ta problem smo reševali s HP filtrom.



Slika 4.3: Frekvenčni spekter spektralnega pošiljanja podatkov

4.4 Tvorjenje zvoka

Tvorjenje zvoka poteka na dva načina. Na računalniku za tvorjenje zvoka uporabljamo program Audacity, ki omogoča enostavno generiranje, združevanje tonov, filtre in predvajanje. Na mobilni aplikaciji pa je bilo potrebno implementirati lastno generiranje zvoka na podlagi fomule (4.1), kjer $freq$ predstavlja frekvenco, ki jo želimo zgenerirati, in s_r predstavlja frekvenco vzorčenja, ki je v našem primeru enaka 44100. Tako pridobljeno vrednost frekvence pomnožimo z vrednostjo, ki je odvisna od tega, kje v tonu se nahajamo. V primeru, da se nahajamo na začetku tona, bomo vrednost postopno zviševali in na ta način poskrbeli, da pri predvajanju zvoka ne bo prišlo do klikajočega zvoka. Na koncu tona bomo tonu glasnost enakomerno zniževali. V sredini tona bomo vrednost pomnožili z maksimalno vrednostjo javanskega razreda Short ter tako poskrbeli za maksimalno vrednost izhoda zvoka. Po končanem množenju naš zvok pretvorimo v zapis Byte, da ga lahko predvajamo v naši aplikaciji.

$$x_n = sin(\frac{2\pi freqi}{s_r}) \tag{4.1}$$

Poglavje 5

Rezultati

V tem poglavju sledi predstavitev rezultatov delovanja naše aplikacije s pomočjo vseh treh implementiranih metod. Preizkusili smo pošiljanje podatkov na dva načina: pri prvem načinu smo za pošiljanje podatkov uporabili računalniške zvočnike, pri drugem pa smo pošiljanje preverjali na mobilnih napravah. Cilj naše aplikacije je bil prenos 32 bitov podatkov v času od dveh do treh sekund.

Pri prvem načinu smo uporabili vse tri metode, torej binarno, spektralno in STMF metodo, pri pošiljanju med mobilnima napravama pa smo pošiljali z binarno in STMF metodo. Ker se prenos podatkov običajno dogaja v okolju, kjer je veliko šumov, zvokov ipd., smo to poskusili posnemati. Celoten spekter kriterijev, ki smo jih testirali:

- kvaliteta zvočnikov;
- dolžina posameznega znaku;
- šum v okolici in
- razdalja do zvočnika.

Pri testiranju smo predpostavili, da je naprava, ki prejema podatke, s svojim mikrofonom obrnjena proti viru podatkov. Pri preverjanju smo merili dve stvari: najvišjo hitrost prenosa podatkov, ki ga lahko pri podani kombinaciji kriterijev dosežemo, in zanesljivost prenosa (torej kolikšen delež

paketov je bil poslan pravilno). Za preverjanje najvišje hitrosti prenosa smo pri vsaki metodi predstavili več podmetod, ki se med seboj razlikujejo v hitrosti pošiljanja znakov. Na ta način smo preverili, na kolikšni razdalji metoda sploh deluje in na kakšen način jo lahko uporabljamo.

Vsako možno kombinacijo kriterijev smo preverili s tridesetimi testi, tako da smo lahko dobili precej dobro sliko, kako metoda sploh deluje na podanih kriterijih.

Naprave, ki smo jih pri testiranjih uporabljali, so bile:

- mobilni telefon LG Nexus 5;
- tablični računalnik Asus Nexus 7;
- namizni računalnik in
- računalniški zvočniki Genius SP-M200.

Zvoke smo predvajali v programu Audacity, saj program omogoča hitro manipulacijo z zvokom v primeru, da jo potrebujemo. Glasnost okolice smo merili z aplikcijo za mobilne telefone Sound Meter PRO.

5.1 Prenos podatkov računalnik - telefon

Pri prenosu podatkov računalnik-telefon smo podatke prenašali s pomočjo vseh treh implementiranih metod. Pri vsaki metodi smo imeli tri ali štiri različne podmetode, ki so se med seboj razlikovale glede na hitrost in količino poslanih podatkov.

Kriterija, ki smo ju uporabljali, sta bila razdalja v centimetrih (kjer smo merili uspešnost na: 0, 20, 40, 60, 80, 100 in 150 centimetrih), ter glasnost okolice (kjer smo s pomočjo belega šuma generirali tri različne glasnosti okolice: 30 dB SPL (tiha soba, šepet), 60 dB SPL (glasnost pogovora) in 80 dB SPL (različna mehanska orodja)). Skupaj smo tako imeli kombinacijo 21 različnih kriterijev.

Binarno prejemanje

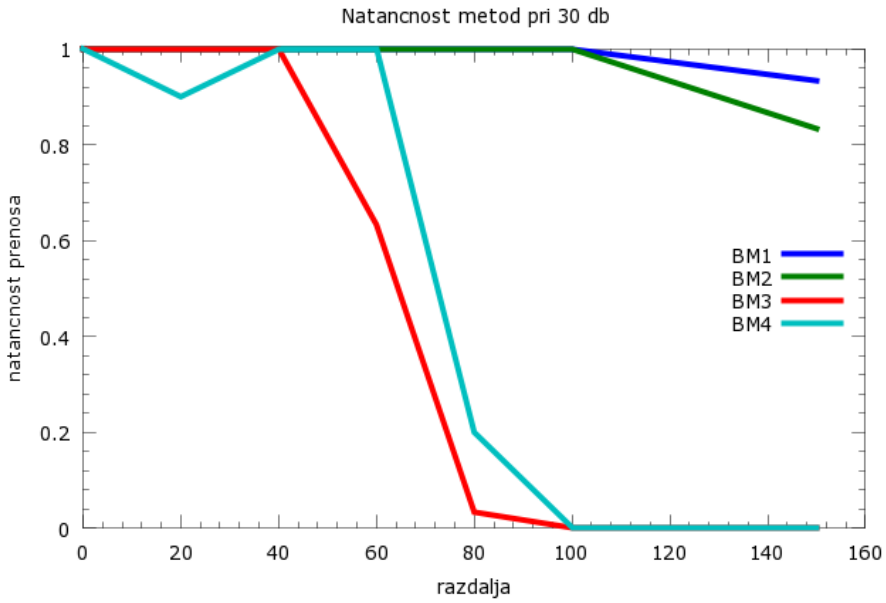
Pri binarnem pošiljanju podatkov smo uporabili štiri metode:

- metoda $BM1$ s hitrostjo 32 bitov v treh sekundah;
- metoda $BM2$ s hitrostjo 72 bitov v treh sekundah;
- metoda $BM3$ s hitrostjo 64 bitov v sekundi in pol in
- metoda $BM4$ s hitrostjo 88 bitov v eni sekundi.

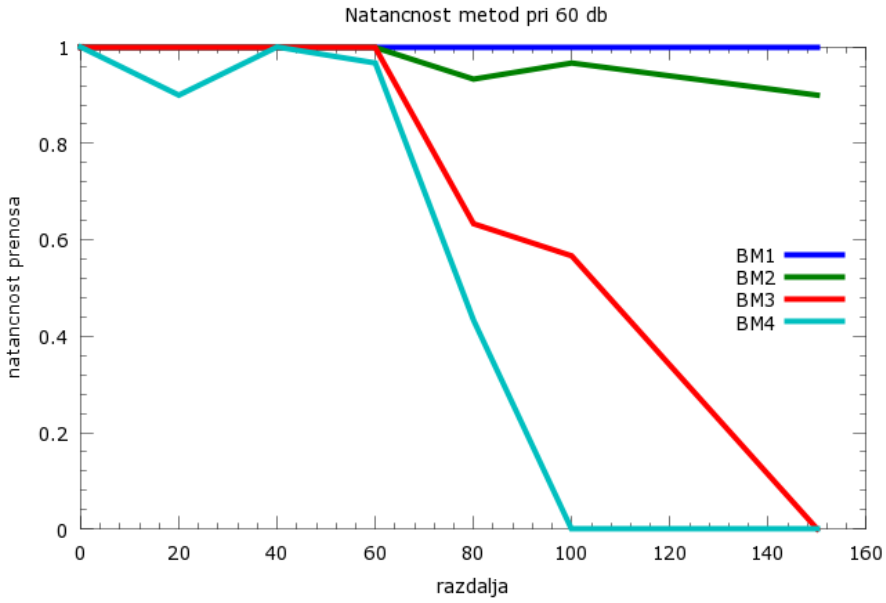
Podatki, ki smo jih pošiljali, so bili 01100110. Te podatke smo si izbrali zato, ker vsebujejo vse možne prehode: prehod iz 0 na 0, iz 0 na 1, iz 1 na 0 in iz 1 na 1. Na podlagi uspešnosti vseh prehodov smo lahko sklepali, da bo prenos uspešen tudi v drugih primerih, ki ne bo uporabljal vseh prehodov.

Vse hitrosti se zaradi varnosti prepolovijo, vendar lahko varnost tudi izklopimo, zato je kot hitrost napisana višja številka. Na prvi pogled nam kot najboljša izgleda metoda $BM4$, saj ima daleč najvišjo hitrost, vendar pa je potrebno omeniti, da postaja zvok vedno bolj slišen, ko se bližamo najhitrejšim prenosom podatkov, saj je vedno težje skrbeti za postopno zviševanje glasnosti.

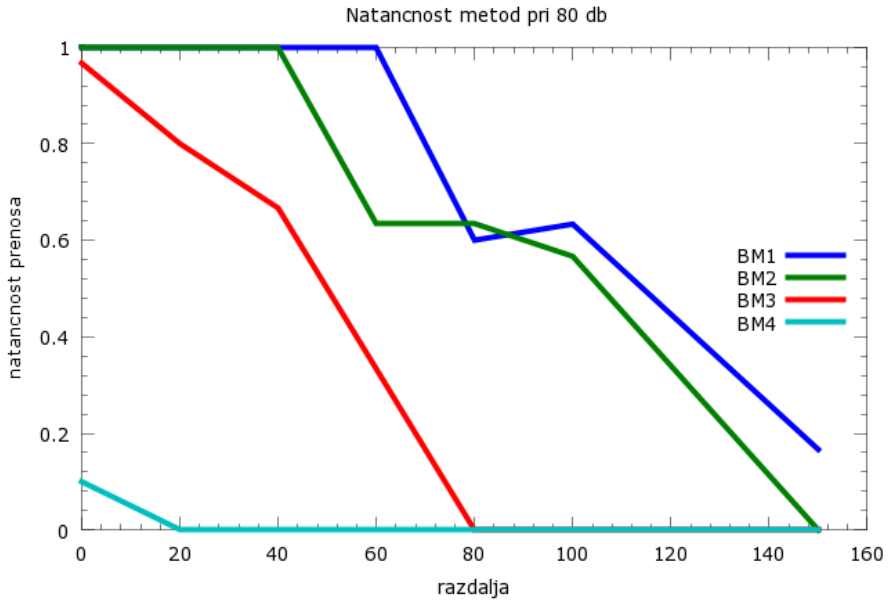
Kot lahko vidimo na grafu 5.1, se natančnost prenosa pri obeh hitrih metodah zelo hitro zniža na nizko stopnjo, medtem ko počasnejše metode ostanejo bolj zanesljive tudi na daljših razdaljah. Pri večjem šumu, kot kaže graf 5.2, se natančnost zniža počasnejšima metodama, medtem ko se, presenetljivo, dvigne pri obeh hitrejših metodah. To si lahko razlagamo tako, da zvok lažje potuje skoz zrak, ki je že delno nasičen z različnim zvočnim valovanjem, ki pa seveda ne sme biti preglasno. Pri 80 dB SPL, kot kaže graf 5.3, se začne večje izgubljanje paketov in s tem tudi močno zmanjšana natančnost pošiljanja. Pri glasnosti višji od 85 dB SPL, nam paketa ni uspelo uspešno poslati z nobeno izmed metod, saj se izgubijo skoraj vsi paketi.



Slika 5.1: Natančnost prenosa vseh štirih binarnih metod pri 30 dB SPL



Slika 5.2: Natančnost prenosa vseh štirih binarnih metod pri 60 dB SPL



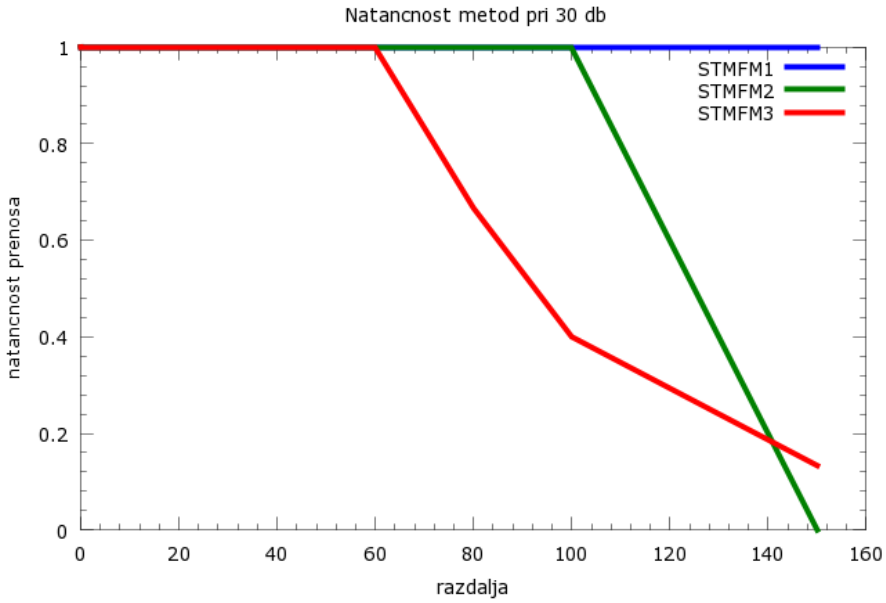
Slika 5.3: Natančnost prenosa vseh štirih binarnih metod pri 80 dB SPL

STMF prejemanje

Pri STMF pošiljanju podatkov smo pošiljali celotno abecedo s pomočjo treh metod. Za abecedo smo privzeli angleško abecedo, ki ima 26 črk. Z vsako črko prejmemo $\lceil log_2 26 \rceil$ bitov, torej pet bitov. V primeru poslane celotne abecede smo prejeli $26 * 5 = 130$ bitov. Uporabljene metode so bile naslednje:

- metoda $STMFM1$ s hitrostjo pet znakov na sekundo $\rightarrow$ 25 b/s;
- metoda $STMFM2$ s hitrostjo sedem znakov na sekundo $\rightarrow$ 35 b/s in
- metoda $STMFM3$ s hitrostjo devet znakov na sekundo $\rightarrow$ 45 b/s.

Podobno kot pri binarnem pošiljanju, se tudi tu pri tišini za razdalje kot bolj zanesljiva izkaže počasnejša metoda (graf 5.4), pri 60 dB SPL šuma (graf 5.5) se natnčnost prenosa izboljša pri hitrih metodah, poslabša pa se pri počasni metodi. Pri najvišji glasnosti (graf 5.6) se natančnost zmanjša, vendar pa se le-ta ne zmanjša tako zelo kot pri binarnem prejemanju, saj smo



Slika 5.4: Natančnost prenosa vseh treh STMF metodah pri 30 dB SPL

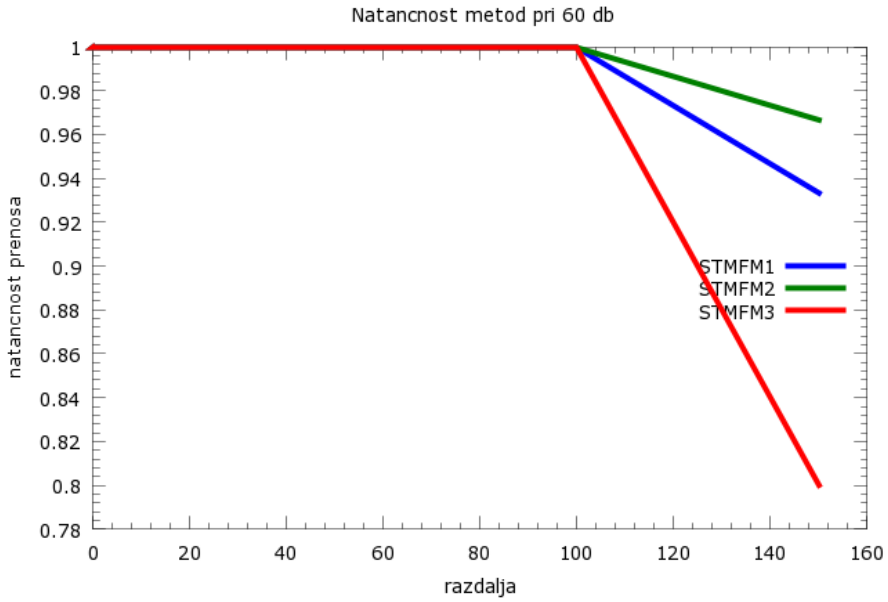
pri vseh metodah še vedno v vsaj nekaj primerih, tudi pri največji razdalji, prejeli pravilne pakete.

Spektralno prejemanje

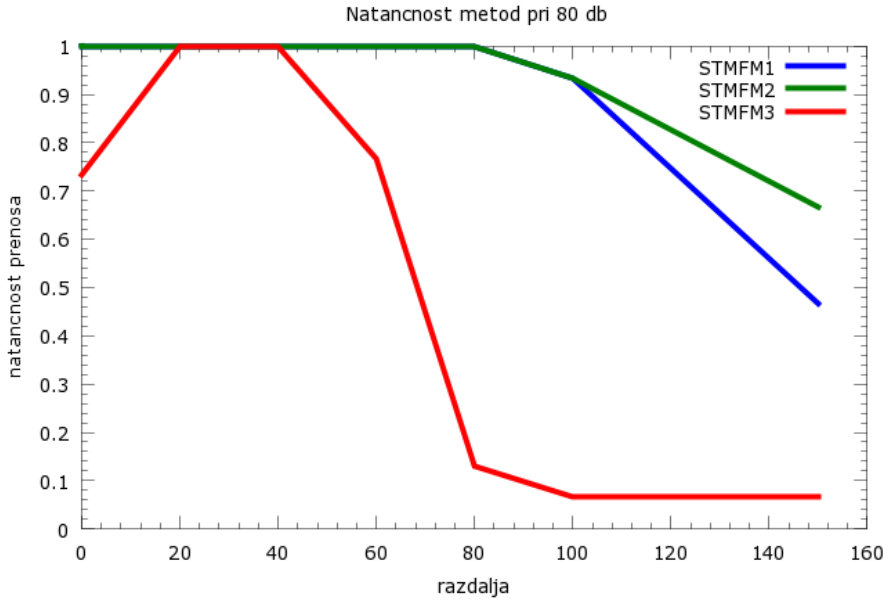
Pri spektralnem pošiljanju zvokov smo uporabili podmetode:

- metoda $SPECM1$, kjer se je vrednost ure spreminjala na četrtinko sekunde $\rightarrow$ 32 bitov na sekundo;
- metoda $SPECM2$, kjer se je vrednost ure spreminjala na tretjino sekunde $\rightarrow$ 24 bitov na sekundo in
- metoda $SPECM3$, kjer se je vrednost ure spreminjala na osmino sekunde $\rightarrow$ 64 bitov na sekundo.

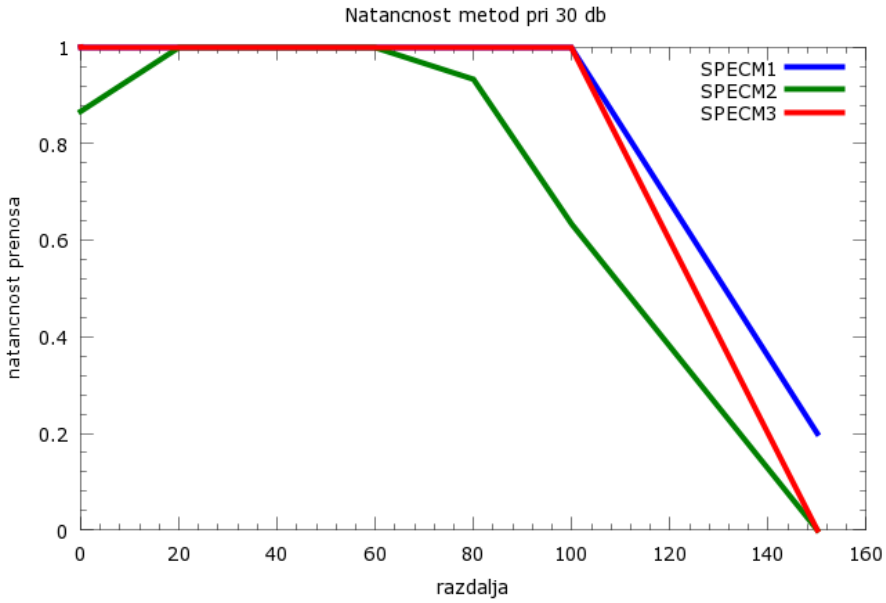
Tudi tu se podobno kot pri binarnem pošiljanju hitrost prepolovi v primeru uporabe varnosti. V primeru spektralnega pošiljanja je to dobra ideja,



Slika 5.5: Natančnost prenosa vseh treh STMF metodah pri 60 dB SPL



Slika 5.6: Natančnost prenosa vseh treh STMF metodah pri 80 dB SPL

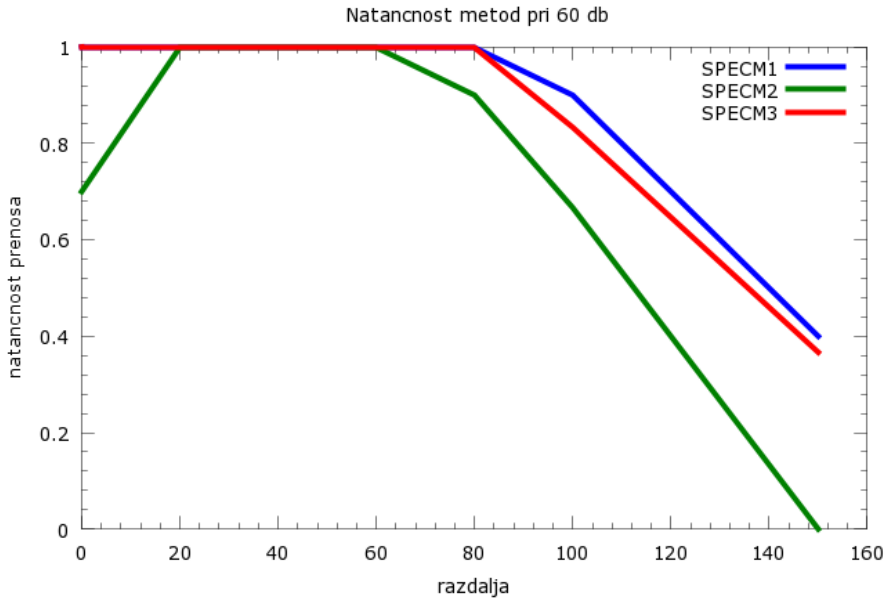


Slika 5.7: Natančnost prenosa vseh treh spektralnih metod pri 30 dB SPL

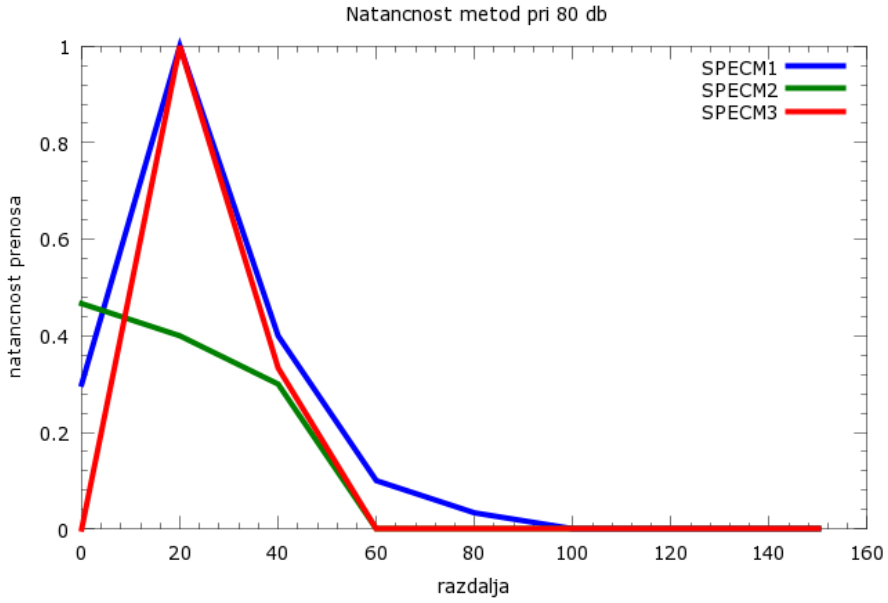
saj se pogosto zgodi, da je narobe poslan le en bit, kar izhaja iz narave prejemanja spektra. Pri spektru je opazno tudi to, da je najpočasnejša metoda pravzaprav tudi najmanj natančna (graf 5.7), kar se precej razlikuje od prejšnjih metod. Tudi pri 60 dB ni slika nič drugačna (graf 5.8), saj najpočasnejša metoda začne prva izgubljati natančnost. Pri najglasnejšem prejemanju (graf 5.9) je opazno tudi to, da je natančnost pri nič centimetrih zelo slaba, in da se prejemanje izboljša šele pri manjši oddaljenosti. To si lahko razlagamo s specifičnostjo pošiljanja spektralnega zvoka.

Primerjava rezultatov in evalvacija

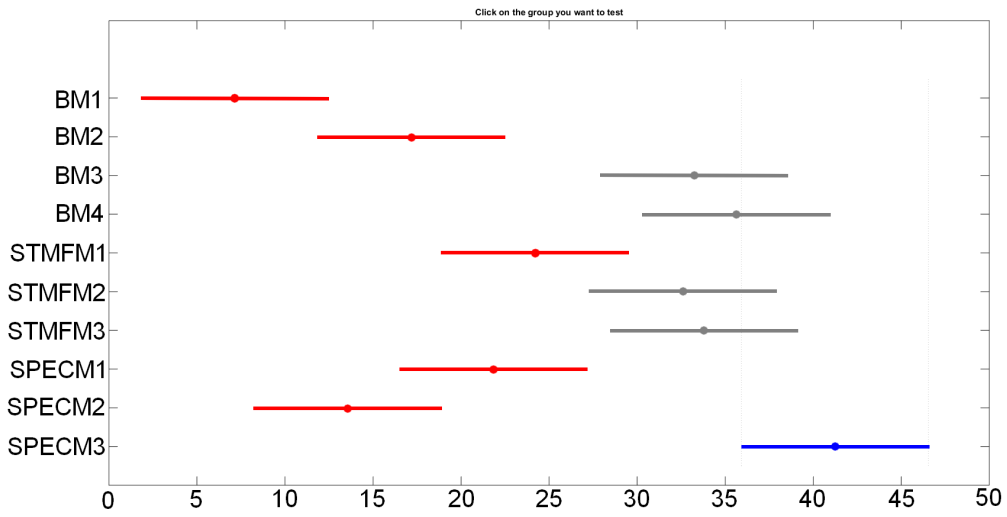
Seveda nas zanima, katera metoda je v našem primeru najboljša za uporabo, kar pa je zelo subjektivno. Nekateri imajo raje natančnejši prenos v zameno za počasno prejemanje, spet drugi želijo hitro prejemanje. V našem primeru smo preverjali, s katero metodo dosežemo najvišji prenos podatkov (torej s katero metodo smo v povprečju poslali največ podatkov), in katera metoda



Slika 5.8: Natančnost prenosa vseh treh spektralnih metod pri 60 dB SPL



Slika 5.9: Natančnost prenosa vseh treh spektralnih metod pri 80 dB SPL



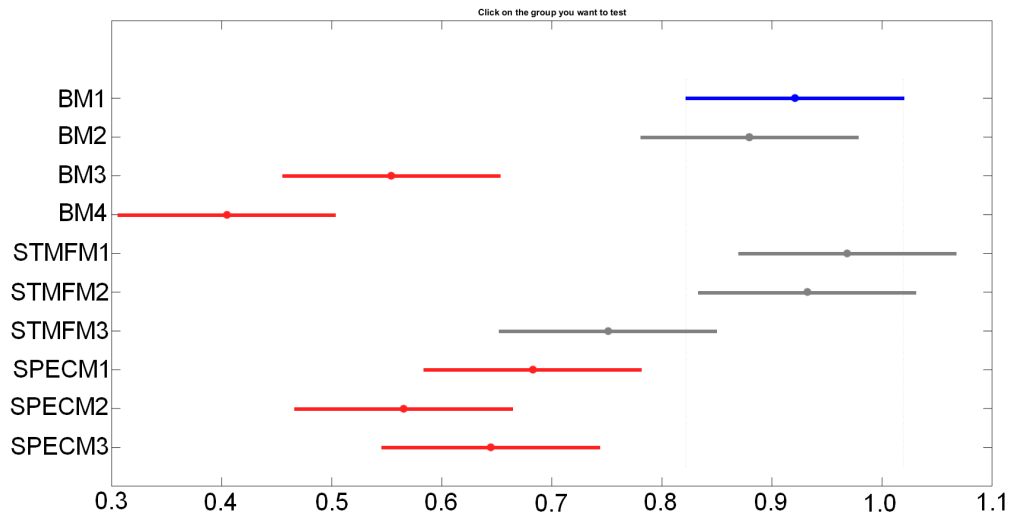
Slika 5.10: Povprečni prenos vseh metod

je najbolj zanesljiva (torej ima najvišji odstotek uspešno prejetih paketov).

Za primerjavo metod smo uporabljali analizo variance, ki jo imamo že implementirano v programu Matlab.

V primeru najvišjega možnega prenosa podatkov smo izračunali natančnost metode in jo ovrednotili s tem, koliko bitov je metoda lahko naenkrat prenesla. Tako je višjo oceno dobila metoda, ki je v petih poskusih trikrat pravilno prenesla 40 bitov, kot pa metoda, ki je petkrat pravilno prenesla 10 bitov. Na grafu 5.10 je lepo vidno, da je ima največji dosežen prenos metoda $SPECM3$, ki ima povprečen prenos okoli 40 bitov. Poleg te metode so med hitrejšimi tudi naslednje: $STMFM2$, $STMFM3$, $BM3$ in $BM4$. To so vse metode, ki so že bile omenjene kot hitrejše. Ostale metode so od omenjenih metod pomembno slabše, kar pomeni, da ne dosegajo dovolj hitrega prenosa podatkov. Vendar pa kot lahko opazimo v prejšnjih grafih (5.1, 5.8, 5.6) izkazujejo metode, ki imajo hiter prenos, precej slabo pošiljanje preko razdalje.

To pa je bil naš naslednji kriterij; kolikšno natančnost naša metoda do-



Slika 5.11: Povprečna natančnost vseh metod

sega. V tem primeru metod nismo ovrednotili s količino bitov. Upoštevali smo le, kolikšna je povprečna natančnost, ki jo je metodi uspelo doseči na vseh možnostih drugih kriterijev (šum in razdalja). Tu je graf skoraj zrcalno obraten prejšnjemu (5.11). Vse metode, ki so bile prej naštete kot najhitrejše pri pošiljanju podatkov, so v tem primeru opazno slabše od najnatančnejših $BM1$, $BM2$, $STMFM1$ in $STMFM2$.

Edina metoda, ki se je znašla tako med najnatančnejšimi kot najhitrejšimi, je metoda $STMF2$. Sklepamo lahko, da je za najbolj optimalno pošiljanje torej najprimernejša ta metoda. Omogoča nam pošiljanje podatkov s hitrostjo 35 bitov na sekundo, kar pomeni, da nam je z najoptimalnejšo metodo uspelo doseči naš cilj.

5.2 Prenos podatkov telefon - telefon

Pri pošiljanju podatkov s telefona na telefon smo uporabljali le binarno in STMF metodo, saj je bila spektralna metoda za mobilne telefone premalo

natančna za kakršno koli uporabo. Pri metodah smo z dolžino znaka določili trajanje. V sekundi predvajanega zvoka je bilo v našem primeru 44100 mest za zapis zvoka. V primeru, da uporabimo dolžino znaka 2000, moramo zraven prišteti še enako dolžino, ki je namenjena tišini med znakoma. Za pošiljanje znaka v dolžini 2000 moramo torej zapisati 4000 mest v našem zvoku.

Pri binarnem pošiljanju je pošiljanje omejeno predvsem s kvaliteto zvočnika, ki ga imamo na voljo. V našem primeru smo uporabljali tablični računalnik Nexsus 7, ki ima serijsko vgrajen malo večji zvočnik. Problem, ki nastane pri predvajanju s serijskimi zvočniki, je ta, da veliko proizvajalcev mobilnih naprav v zvočnike vgrajuje ojačevalce zvoka. Težava pri ojačitvi zvoka je ta, da pri njej nastajajo frekvence, ki so slišne.

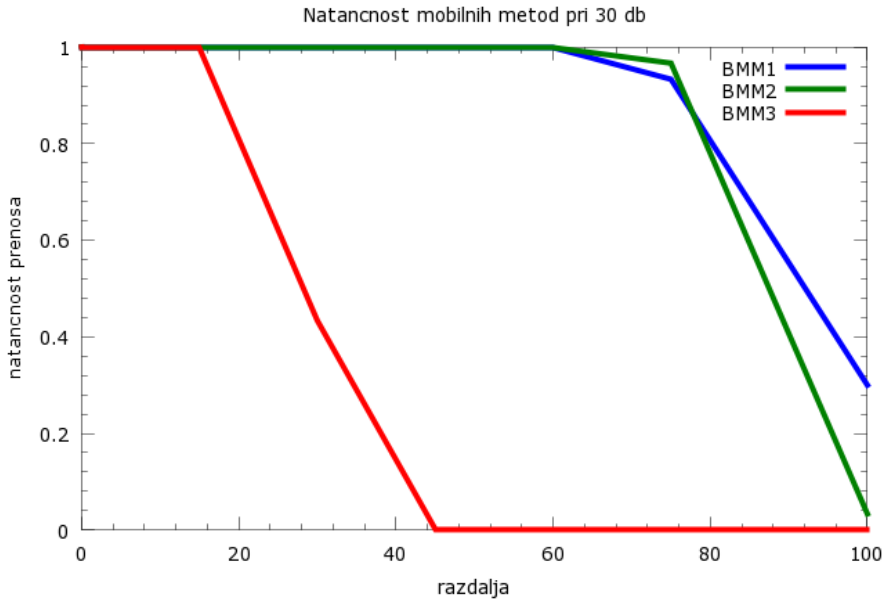
Pri prenosu preko mobilnih naprav smo razdalje primerno zmanjšali, saj ni imelo smisla pričakovati možnosti pošiljanja na takšne razdalje kot pri pošiljanju iz računalnika. Uspešnost prejema smo merili na 0, 5, 15, 30, 45, 60, 75 in 100 centimetrih. Glasnost smo merili na treh nivojih: 30 dB SPL, 60 dB SPL in 80 dB SPL.

Binarno prejemanje

Pri binarnem pošiljanju smo uporabili tri metode:

- metoda $MBM1$ z dolžino znaka 2000 $\rightarrow$ 11 bitov na sekundo;
- metoda $MBM2$ z dolžino znaka 1000 $\rightarrow$ 22 bitov na sekundo in
- metoda $MBM3$ z dolžino znaka 500 $\rightarrow$ 44 bitov na sekundo.

Tudi v primeru mobilnega prenosa zvokov smo pri binarnem pošiljanju uporabljali besedo 01100110. Razloge za to smo navedli že prej. V primeru mobilnega telefona smo pričakovali, da bo šum precej bolj vplival na prenos podatkov, saj se zvočnik mobilnega telefona nikakor ne more meriti z računalniškim zvočnikom. Vendar pa je šum v našem primeru vplival podobno kot na računalniške zvočnike, kar lahko vidimo tudi na grafu 5.12. Natančnost prejemanja se je v primeru 60 dB SPL ponovno dvignila (graf



Slika 5.12: Natančnost prenosa vseh treh binarnih metod pri 30 dB SPL

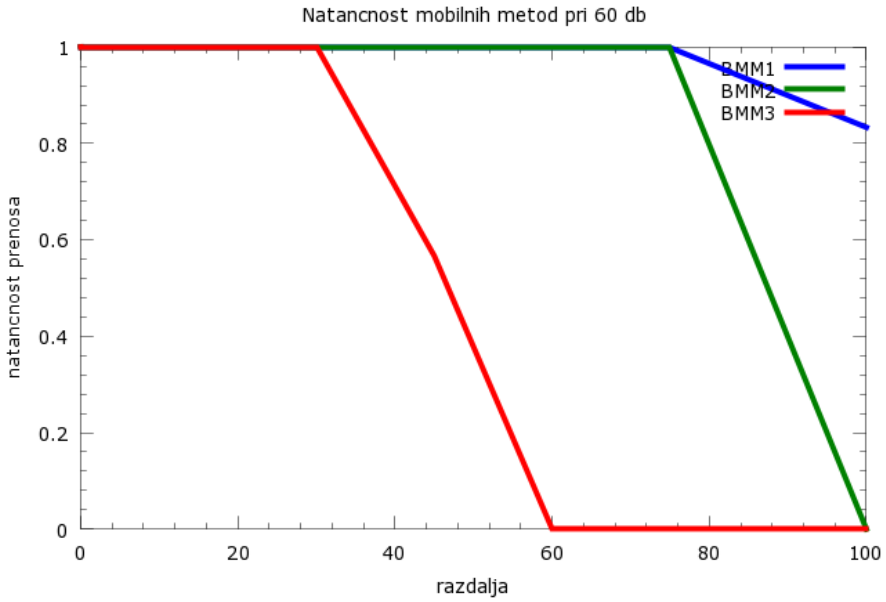
5.13), kar je najbolj opazno pri najhitrejši metodi. Glasnost pa je začela vplivati po 75 dB SPL, ko je začela povzročati izgube (graf 5.14).

STMF prejemanje

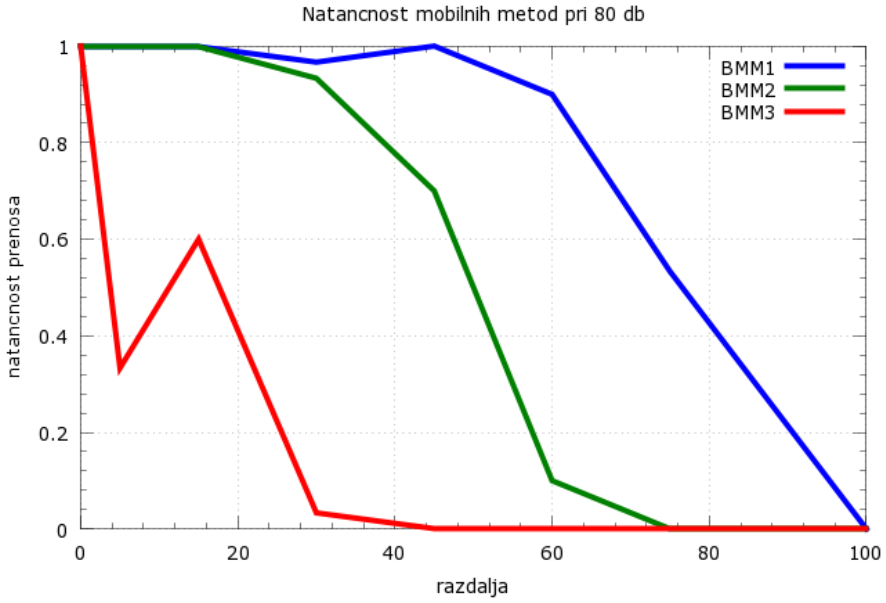
Pri STMF pošiljanju smo preizkusili tri metode, vendar pa so bili rezultati pri dolžini znaka 1500 tako slabi, da jih nismo uporabili. Tako smo na koncu uporabili dve metodi:

- metodo $MSTMFM1$ z dolžino znaka 2500 $\rightarrow$ 8 znakov na sekundo $\rightarrow$ 40 bitov na sekundo in
- metodo $MSTMFM2$ z dolžino znaka 2000 $\rightarrow$ 11 znakov na sekundo $\rightarrow$ 55 bitov na sekundo.

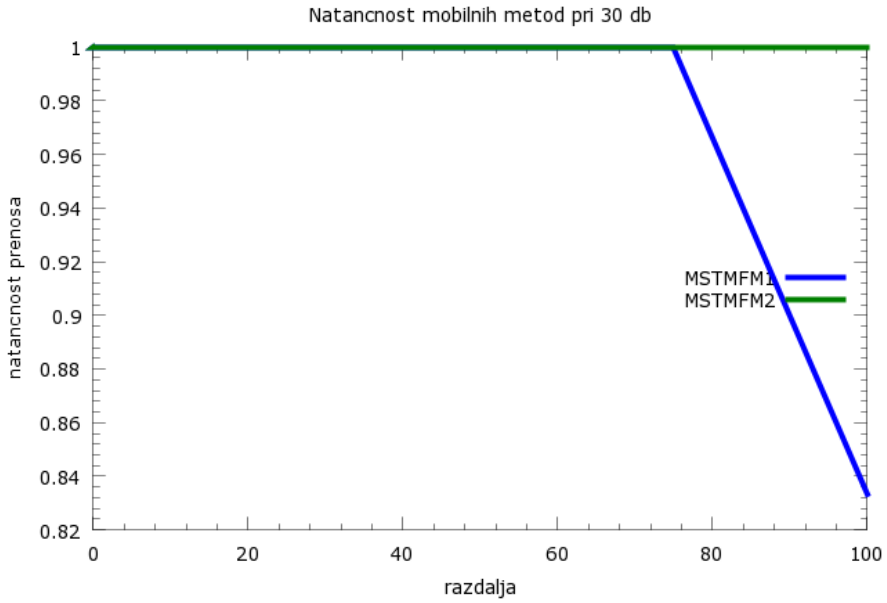
V primeru STMF pošiljanja smo ponovno pošiljali celotno abecedo. Tako smo se lahko prepričali, da pošiljanje deluje na vseh črkah in ne le na izbranih. Zaradi večje informacijske vrednosti podatkov smo si lahko privoščili



Slika 5.13: Natančnost prenosa vseh treh binarnih metod pri 60 dB SPL



Slika 5.14: Natančnost prenosa vseh treh binarnih metod pri 80 dB SPL



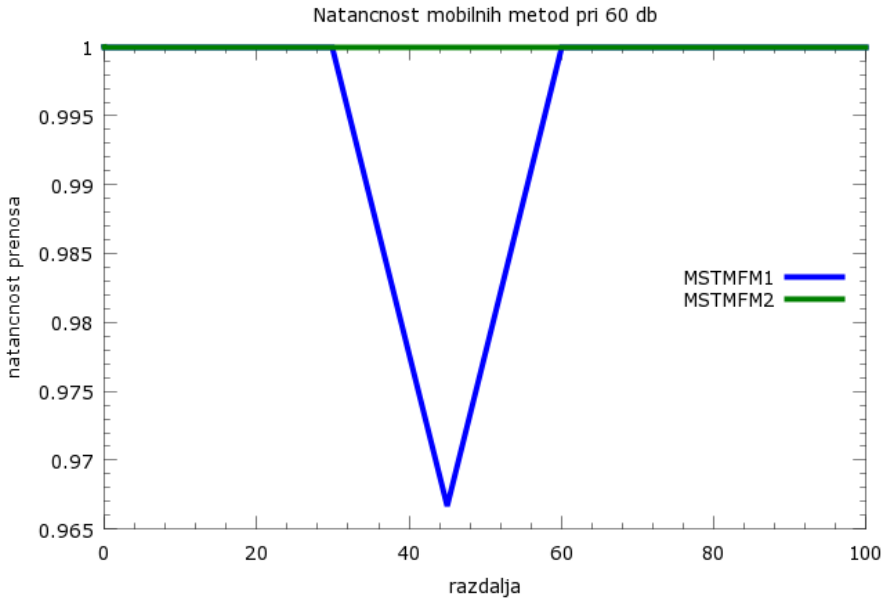
Slika 5.15: Natančnost prenosa vseh treh STMF metod pri 30 dB SPL

počasnejše pošiljanje kot v binarnem primeru. Kasneje smo opazili tudi to, da se pri poskusu hitrejšega pošiljanja (1500) paketi začnejo izgubljati že pri najtišjih glasnostih. Preostali metodi imata zaradi svoje počasnosti precej večji domet, zato je pri najtišjih pošiljanjih prenos dokaj dober (graf 5.15). Tudi glasnejše okolje ne prepreči relativno dobrega prejemanja (graf 5.16), saj tudi pri največji glasnosti dosežemo več kot polovico pravilnih prenosov na najdaljši razdalji (graf 5.17).

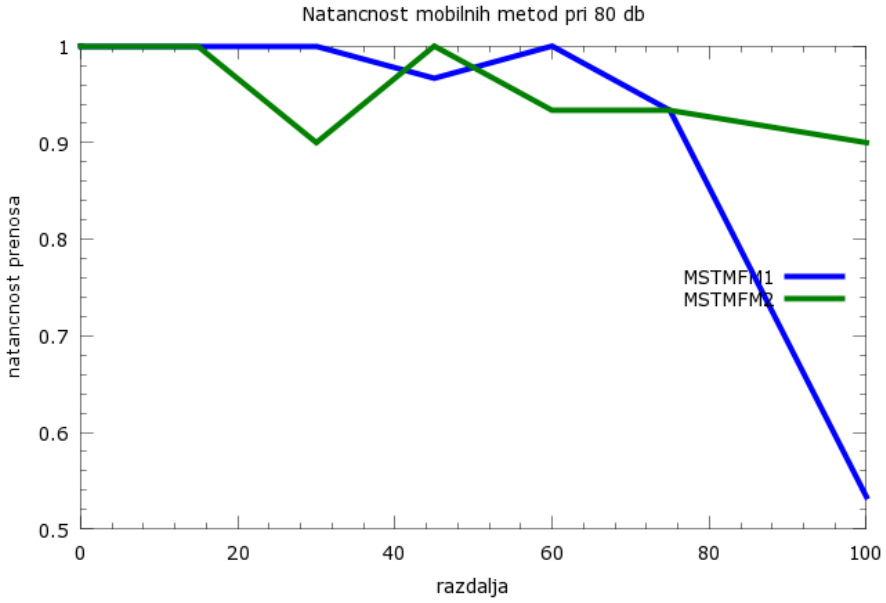
Primerjava rezultatov in evalvacija

V primeru pošiljanja podatkov preko mobilne naprave je odločitev o najboljši metodi precej lažja, kot pri pošiljanju preko računalnika, saj imamo zaradi manjšega števila metod lepši graf multivariančne analize.

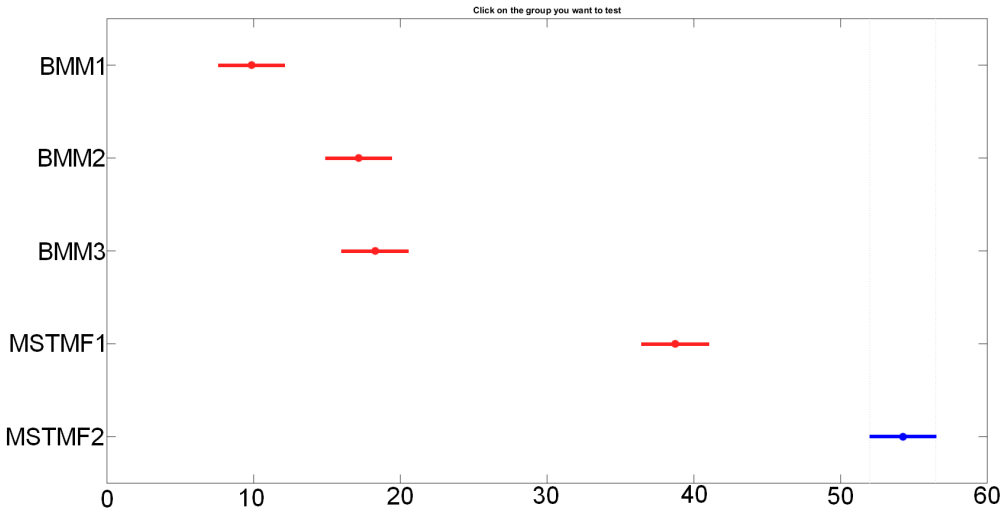
V primeru najvišjega prenosa podatkov je odločitev popolnoma nedvoumna, saj je metoda $MSTMF2$ boljša od vseh ostalih metod. To je vidno tako, da se repi metode, ki predstavljajo njen odklon, ne prekrivajo s katero



Slika 5.16: Natančnost prenosa vseh treh STMF metod pri 60 dB SPL



Slika 5.17: Natančnost prenosa vseh treh STMF metod pri 80 dB SPL



Slika 5.18: Povprečni prenos vseh mobilnih metod

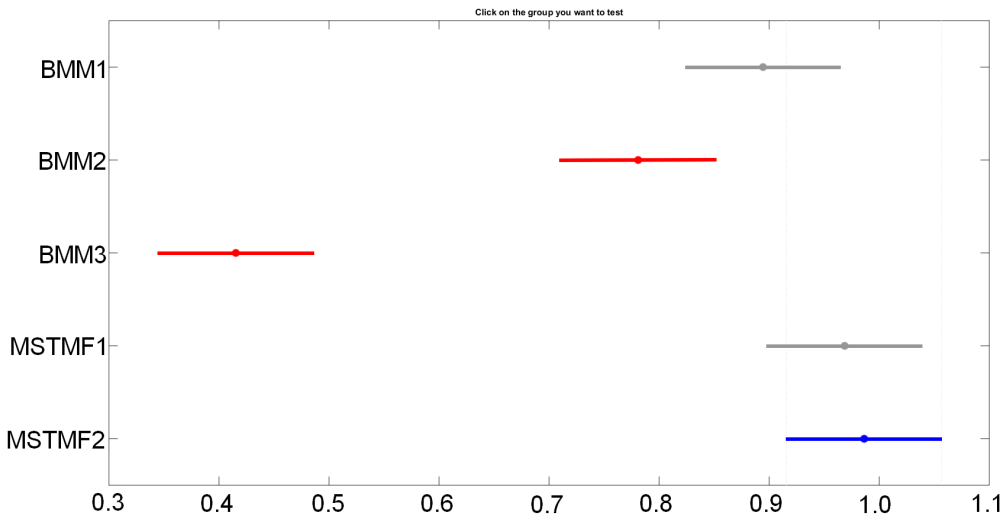
koli drugo metodo (graf 5.18).

Kar se tiče natančnosti, se graf 5.19 malenkostno spremeni, tako da med najnatančnejše metode spadajo $BMM1$, $MSTMF1$ in $MSTMF2$. Vendar pa je zmagovalec mobilnega prejemanja popolnoma jasen, saj se metoda $MSTMF2$ pojavi v obeh oblikah.

Najoptimalnejše prejemanje pri pošiljanju podatkov iz mobitela na mobitel je tako metoda $MSTMF2$, ki ima hitrost prenosa 55 bitov na sekundo, kar pomeni, da smo pri pošiljanju podatkov preko mobilnih naprav dosegli svoj cilj.

5.3 Sklepi

Ne glede na to, ali bomo pošiljali z binarno, STMF ali spektralno metodo in ali bomo pošiljali z mobilno napravo ali računalnikom, bomo pri pošiljanjih prisiljeni sklepati kompromise. V našem primeru sklepamo kompromise med natančnostjo prejemanja, ki je pogojena z šumom okolice, in razdaljo od



Slika 5.19: Povprečna natančnost vseh mobilnih metod

vira zvoka, kvaliteto oddajnika zvoka, ipd., ter hitrostjo prejemanja, ki je pogojena z metodo, ki jo pošiljatelj uporablja, dolžino znakov ter pavz med znaki. Odločitev o pomembnosti vsakega parametra je seveda v odločitvi vsakega posameznika. V primeru, da bo prenos vedno potekal na majhnji razdalji, se lahko hitrost poveča, v primeru pošiljanja na malo daljše razdalje, pa se lahko hitrost poljubno zmanjša.

Kriterij, ki se pri prejemanju podatkov ni izkazal za pomembnejšega, je bila okenska funkcija. Okenska funkcija je sicer pomembna, saj se v primeru prejemanja podatkov brez nje natančnost občutno zniža. Ni pa pomembno, katero vrsto okenske funkcije vzamemo, saj so imele vse tri uporabljene podobne rezultate. Varnost je v našem primeru veliko bolj potrebna v primeru spektralnega prejemanja, saj se pri binarnem redko zgodi enojna napaka. Pri binarnem prejemanju se običajno v primeru napake biti izgubijo. V tem primeru nam naša varnost ne pomaga. Pomembna kriterija v našem primeru sta šum okolice in razdalja prejemanja, kar se precej dobro vidi na grafih, ki prikazujejo natančnost prejemanj pri različnih šumih in razdaljah (5.1, 5.13,

5.6).

Uspešnost prenosa na daljše razdalje lahko pri računalniku zelo preprosto povečamo, saj moramo le nadgraditi zvočnike, ki so sposobni boljšega in glasnejšega pošiljanja zvoka. V primeru mobilnega telefona, pa je to težje, saj notranjih zvočnikov ne moremo zamenjati. Težavo bi lahko delno rešili s pomočjo zunanjih zvočnikov.

Za cilj smo si zadali prenos 32 bitov v času od dveh do treh sekund. To nam je pri vseh metodah uspelo, zato lahko rečemo, da smo cilj uspešno dosegli. Katera metoda je najboljša za pošiljanje podatkov, je težko trditi, saj ima vsaka metoda svoje pluse in minuse. STMF metoda lahko pošlje večjo količino informacij, vendar pa nima dodane varnosti, zato napak ne moremo popravljati. Spektralna metoda ne vsebuje pavz med znaki, zato lahko dosežemo višjo hitrost prenosa, vendar pa ima na razdalji slab prenos, večkrat pa se pojavijo tudi napake. Pri binarnem zvoku nam pretirano krajšanje zvoka povzroči bližanje slišnim frekvencam. Iz napisanega je razvidno, da ne moremo z gotovostjo trditi katera je metoda boljša, saj ima vsaka svojo optimalno okoliščino, v kateri bi jo lahko uporabljali.

Poglavje 6

Zaključek in nadaljnje delo

V diplomskem delu smo izdelali Android aplikacijo, katere namen je prejemanje in pošiljanje podatkov s frekvencami, ki so človeku neslišne. Pri tem smo uporabljali frekvence, ki spadajo v frekvenčni prostor med 16 in 20 kHz. Za prejemanje podatkov smo implementirali tri metode: binarno, STMF in spektralno. Pri binarnem in spektralnem pošiljanju smo dodali varnost v obliki Hammingovega koda. V primeru STMF pošiljanja pa lahko pošljemo več kot le nič in ena.

Cilj naše aplikacije je bil prenos 32 bitov v času od dveh do treh sekund. To nam je uspelo z vsemi tremi metodami, pravzaprav smo pri vseh metodah dosegli prenos podatkov višji oz. hitrejši od našega cilja. Najhitrejši prenos nam je seveda uspelo doseči v idealnih razmerah, kjer se niso upoštevali različni kriteriji, ki jih srečamo v vsakdanjem življenju, kot so šum okolice, razdalja prenosa ipd. Pri vsaki metodi smo preizkusili več različnih podmetod, ki so podatke pošiljale z različno hitrostjo. Vse naše podmetode smo skupaj z vsemi možnimi kombinacijami kriterijev preizkusili na veliko testih. Na ta način smo si lahko ustvarili sliko, kako učinkovita je sploh neka metoda za pošiljanje in prejemanje podatkov. Nad vsemi podatki smo izvedli analizo variance in tako izvedeli, katere metode imajo najvišji prenos podatkov ter katere metode imajo višjo natančnost. Izbrali smo tudi optimalne metode pri obeh vrstah pošiljanja.

Razvoj aplikacije gre lahko naprej v smeri dinamičnega prilagajanja velikosti okna, ki zajema zvok za obdelavo. Na ta način bi lahko pri določeni hitrosti in razdalji uporabili okno, ki bi bilo za ta namen bolj primerno. To pa bi prineslo tudi dodatne težave, saj bi bilo potrebno napravi pred prejemanjem sinhronizirati. Pri varnosti bi lahko namesto Hammingovega koda, ki lahko popravlja le eno napako, implementirali algoritme, ki lahko zaznavajo tudi izbruh napak in jih uspešno popravijo.

Aplikacijo bi bilo možno uporabljati vsakodnevno, pri različnih preprostih nalogah, npr. pri prenosu naše vizitke na drug telefon ali ko bi šli mimo izložbe in bi nam ta sporočila, kakšno je danes vreme ipd. Prenos nikakor ni namenjen prenosu občutljivih podatkov, saj nam lahko prisluškuje vsak, ki na podoben način dekodira podatke.

Literatura

[1] S. Rosen and P. Howell, *Signals and systems for speech and hearing*, vol. 29. Brill, 2011.

[2] D. R. Smith, *Digital transmission systems*. Springer Science & Business Media, 2012.

[3] W. E. Company, *Fundamentals of Telephone Communication Systems*. 1969.

[4] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 4, pp. 23–33, 2007.

[5] "Mobile acoustic modems in action." `https://code.google.com/p/mobile-acoustic-modems-in-action/wiki/ProjectReport#Conclusions_&_Future_Work`. Accessed: 2015-08-09.

[6] A. Madhavapeddy, R. Sharp, D. Scott, and A. Tse, "Audio networking: the forgotten wireless technology," *Pervasive Computing, IEEE*, vol. 4, no. 3, pp. 55–60, 2005.

[7] "Chirp." `http://chirp.io/faq/`. Accessed: 2015-08-09.

[8] "Nearbytes." `http://nearbytes.com/en/technology.php`. Accessed: 2015-08-09.

[9] V. Filonenko, C. Cullen, and J. Carswell, "Investigating ultrasonic positioning on mobile phones," in *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, pp. 1–8, IEEE, 2010.

[10] W. A. Arentz and U. Bandara, "Near ultrasonic directional data transfer for modern smartphones," in *Proceedings of the 13th international conference on Ubiquitous computing*, pp. 481–482, ACM, 2011.

[11] "Ultrasonic networking on the web." `http://smus.com/ultrasonic-networking/`. Accessed: 2015-08-09.

[12] "Transferring data between mobile devices using ultrasound." `http://rnd.azoft.com/mobile-app-transering-data-using-ultrasound/`. Accessed: 2015-08-09.

[13] "Signal360." `http://www.signal360.com/#contact`. Accessed: 2015-08-09.

[14] "Notifly." `https://www.notifly.eu/`. Accessed: 2015-08-09.

[15] S. W. Smith *et al.*, "The scientist and engineer's guide to digital signal processing," 1997.

[16] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[17] "The scientist and engineer's guide to digital signal processing." `http://www.dspguide.com/ch12/2.htm`. Accessed: 2015-09-09.

[18] J. W. Cooley, P. A. Lewis, and P. D. Welch, "The fast fourier transform and its applications," *Education, IEEE Transactions on*, vol. 12, no. 1, pp. 27–34, 1969.

[19] K. Prabhu, *Window functions and their applications in Signal Processing*. CRC Press, 2013.

[20] D. G. Luenberger, *Information Science*. Princeton University Press, 2006.